Universidad
Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

Predictive Statistical User Models under the Collaborative Approach

Autor:

Leonardo Castaño Zabaleta

Director/es:

Francisco Javier Calle Gómez

Elena Castro Galán

Departamento de informática

Leganés, Julio 2016

TESIS DOCTORAL

Predictive Statistical User Models under the Collaborative Approach

Autor: *Leonardo Castaño Zabaleta*

Director/es: **Francisco Javier Calle Gómez**

Elena Castro Galán

Firma del Tribunal Calificador:

		Firma
Presidente:	Antonio de Amescua Seco	
Vocal:	Ruth Cobos Pérez	
Secretario:	Dominikus Heckmann	
Calificación:		

Leganés, de

Acknowledgments

Though only my name will remain on the cover of this dissertation, many people have contributed to its production. I owe special gratitude and mention for all of them who made this dissertation possible. I want to express my special gratitude to my supervisors, Dr Francisco Javier Calle and Dr Elena Castro Galan whose expertise, understanding and patience have been crucial on the development of this thesis. I appreciate their vast knowledge and skills in areas such as user modelling, recommender systems, machine learning etc. has added a massive value to this thesis. In addition, a very special thanks goes to Dr Dolores Cuadra who despite not being one of the supervisors provided knowledge, motivation and encouragement along the whole thesis. Their dedication and persistence is worth of admiration. They all make a difference in my life developing a real focus on user modelling, recommendation algorithms and data manipulation. That focus has now turned into a passion that I do every day with special gratitude.

I also want to thank my family for their support they provided through my entire life and in particular along the development of this thesis. Special mention goes to Rocio as this thesis would not be possible without her patience love and generosity. She not only provided the right support in the right moments but also helped me overcome setbacks and put me back on the right way. I also want to dedicate this work and all the effort to the memory of my uncle who wisely guided me through my academic career.

I´m also very grateful to the following former or current staff of the University Carlos III, Dr Alejandro Calderon, Pablo Jose Izquierdo and Alejandro Baldominos. Special mention to the latest one for his contribution on the development of this thesis. I am also grateful to many friends who helped me to stay calm and focus during these difficult years.

Additionally, a big thanks to the staff of DFKI and University of Maryland with special mention to Dr Heckmann and Dr Findlater for accepting me in their research groups and provide a different angle to the thesis.

Finally, this thesis would not have been possible without the financial support of the following research projects Cadooh (TSI-020302-2011-21), Thuban (TIN2008-02711) that funded part of this research.

In conclusion, I believe that this research is the sum of a very big human team plus the financial support and every single one contribute with a small but important bit. To all of them a deep thanks for all your effort help and support through this bumpy road.

Summary

User models and recommender systems due to their similarity can be considered the same thing except from the use that we make of them. Both have their root in multiple disciplines such as information retrieval or machine learning among others. The impact has grown rapidly with the importance of data on systems and applications. Most of the big companies employ one of the other for different reasons such as: gathering more customers, boost sales or increase revenue. Thus very well-known companies like Amazon, EBay or Google use models to improve their businesses. In fact, as data becomes more and more important for companies, universities and people, user models are crucial to make decisions over large amounts of data. Although user models can provide accurate predictions on large populations their use and application is not restricted to predictions but can be extended to selection of dialogue strategies or detection of communities within complex domains.

After a deep review of the existing literature, it was found that there is a lack of statistical user models based on experience plus the existing models in the area are content-based models that suffer from major problems as scalability, cold-start or new user problem. Furthermore, researchers in the area of user modelling usually develop their own models and then perform ad-hoc evaluations that are not replicable and therefore not comparable. The lack of a complete framework for evaluation makes very difficult to compare results across models and domains.

There are two main approaches to build a user model or recommender system: the content based approach, where predictions are based on the same user past behaviours; and the collaborative approach where predictions rely on like-minded people. Both approaches have advantages but also downsides that have to be considered before building a model. The main goal of this thesis is to develop a hybrid user model that takes the strengths of both approaches and mitigates the downsides by combining both methods. The proposed hybrid model is based on an R-Tree structure. The selection of this structure to support the models is backed from the fact that the rectangle tree is specifically designed to effectively store and manipulate multidimensional data. This data structure introduced by Guttman in 1984 is a height balanced tree that only requires visiting a few nodes to perform a tree search. As a result, it can manage large populations of data efficiently as only a few nodes are visited during the inference. R-Tree has two different typologies of nodes: the leaf-node and the non-leaf node. Leaf nodes contain the whole universe of users while non leaf nodes are somehow redundant and contain summaries of child nodes.

Along this thesis two statistical user models based on experience have been proposed. The first one is a knowledge base user mode (KLUM), is a classical approach that summarizes and remove data in order to keep performance level within reasonable margins. The second one, an R-Tree user model (RTUM), is an innovative model based on an R-Tree structure. This new model not only solves the problem of removing data but also the scalability problem which turns out to be one of the major problems in the area of user modelling. Both models have been developed and tested with equivalent formulations to make comparisons relevant. Both models are prepared to create their own knowledge base from scratch but also they can be fed with expert knowledge. Thus alleviating another major problem in the area of user modelling as it is the start-up problem.

Regarding the proposal of this thesis, two statistical user models are proposed (KLUM and RTUM). In addition, a refinement of RTUM user model is proposed, while RTUM performs node partitions based on the centroids of the users in that node, the new refinement implements a new partition based on privileged features. Hence, the new approach takes advantage of most discriminatory features of the domain to perform the partition. This new approach not only provides accurate inferences, but also an excellent clustering that can be useful in many different scenarios. For instance, this clustering can be employed in the area of social networks to detect communities within the social network. This is a tough task that has been one of the goals of many researchers during the last few years.

This thesis also provides a complete evaluation of the models with a great diversity of parameterizations and domains. The models are tested in four different domains and as a result of the evaluation, it is proved that RTUM user model provides a massive gain against classical user models as KLUM. During the evaluation, RTUM reached success rates of 85% while the analogous KLUM could only reach a 65% thus leaving a 20% gain for the proposed model. The evaluation provided not only compares models and success rates, but also provides a broad analysis of how every parameter of the models impact the performance plus a complete study of the databases sizes and inference times for the models. The main conclusion to the evaluation is that after a complete evaluation with a wide diversity of parameters and domains RTUM outperforms KLUM on every scenario tested.

As previously mentioned, after the literature review it was also found a lack of evaluation frameworks for user modelling. This thesis also provides a complete evaluation framework for user modelling. This fills a gap in the literature as well as makes the evaluation replicable and therefore comparable. Along years researchers and developers had found difficulties to compare evaluations and measure the quality of their models in different domains due to the lack of an evaluation standard. The evaluation framework presented in this thesis covers data samples including training set and test set plus different sets of experiments alongside with a statistical analysis of the domain, confidence intervals and confidence levels to guarantee that each experiment is statistically significant. The evaluation framework can be downloaded and then used to complete evaluations and cross-validate results across different models.

List of Figures

CONTENTS

1 Introduction

Nowadays most applications and systems use either a recommender system or a user model. In fact, very well-known sites such as: Amazon[1], Ebay[2] or MovieFinder[3] make use of these systems to personalize content and to provide targeted products to their users. Even though, each site has a different implementation of these systems, all of them make use of these systems. Therefore, since online E-Commerce sites until desktops applications need personalization and adaptation due to the vast amount of electronic data and the each time higher expectations of their users. Despite the fact that some time ago systems and applications used to have the same behaviour regardless the user, this trend is over and new trend leads to develop systems to work as if they were designed to fit the requirements of every single user.

User modelling is a multi-disciplinary research topic that belongs to the following areas, Human-Computer Interaction (HCI), Artificial Intelligence and Machine Learning. User modelling discipline focuses on building systems able of adapting their interaction based on the user's features, behaviours, needs, likes and preferences. Thus, the systems must have a representation of the user in order to provide the adaptation effect. Such representation of the universe of users is called user model. Therefore, a user model can be described as a representation of the users that captures the goals of the user that is what the user is attempting to achieve: experiences, interests, likes, preferences and behaviours. The term user model should not be confused with the term user profile, which is usually restricted to user-specified information, disregarding the information acquired about the user by any other means.

Usually, main steps in user modelling involve: choosing the domain and performing domain analysis before building the model. Though this is the classic and most widespread approach, this thesis deals with another approach where user models are domain-independent. Models built under this approach can be reused in different domains. Both the structure and the implementation of the reasoning mechanisms are common and reusable, and even the content acquired through experience is still valid through different domains. Another important distinction among user models is the adaptation focus. In this sense, two main approaches have to be mentioned: the content-based approach and the collaborative approach. The first performs adaptation or produce recommendations for current user based on the preferences and likes of that user. On the other hand, collaborative filtering produces recommendations for current user based on the likes and preferences of other like-minded people.

1.1 Motivation

Long ago developers began struggling for personalization and adaptation in their systems. These provide features which have proved their flexibility to adapt contents to the user needs. In fact, up to day most systems, applications and services employ a user model or a recommender system

---

[1] Amazon: http://www.amazon.com/

[2] Ebay: http://www.ebay.com/

[3] MovieFinder: http://www.moviefinderonline.com/

to behave according to the user needs. The terms user model and recommender system have been used together to refer to the same concept. However, the main difference between recommender systems and user models is the use of the system or the purpose of the system. Usually recommender systems focus on providing recommendations to their users, these recommendations are based on the user's likes and preferences. On the other hand, user models not only provide recommendations, but also seek for adapting the behaviour of the system to be more useful and to fit the specific background of the user. In fact, developers of Human-Computer Interaction systems face the arduous task of designing and writing software for millions of people while making it works as if it was designed for every single user.

In fact, as computers and mobile devices have spread, the software has also become more interactive. Instead of software for masses, new trends points out that every single piece of software must be conceived to fit the requirements of every single user. This new trend magnifies the important of adaptation and personalization in software development. Human-Computer Interaction studies interactions and relationships between humans and computers. Though, HCI is a multidisciplinary area (including User Modelling), most researchers have shown their interest in User Modelling because of the potential of this area to improve the relation between humans and computers.

This thesis belongs to the area of Human Computer Interaction and more precisely to the area of Statistical User Modelling. Therefore, this thesis addresses the problem of providing predictions for unobserved features employing observed features to perform those predictions. As stated above, user modelling is an emergent area where many challenges still have to be addressed. Thus, this thesis tries to deal with main problems in the area of user modelling such as: scalability problem, knowledge loss, response time or the lack of standardized evaluation tools adapted to the specific problem of user modelling.

1.2 Objectives

The research problem addressed on this thesis is the problem of predictive and statistical user models under the collaborative approach. This research problem involves improving the capabilities of software by providing adaptation and personalization. To deal with this problem a new statistical user model approach is proposed and developed. Therefore, there are several thesis goals that will be explained next. Some of them are requirements that the solution proposed has to meet and some of them are new research lines that come to alleviate main barriers in the area of statistical user modelling.

- <u>Baseline model versus new approaches</u>: Main goal of this thesis focuses on developing a new statistical user model that fixes or alleviates main problems of classical user modelling. This goal itself requires develop a baseline user model based on the classical approach to establish a comparison between the new approach and the classical one.
- <u>Response time</u>: Response times lower than one second are considered to be almost real time. The system must provide real time responses to the users. However, offline tasks do not have to meet this requirement.

- Scalability: One of the bigger problems of user modelling is the scalability problem. At the beginning most systems are able to provide quick response times. As systems knowledge bases grow and degenerate, response times trend to grow and fail to complete tasks in bounded response times.
- Domain-Independent: Though, main approach to build user modelling systems involves create a domain-oriented model, this thesis seeks building a domain-independent user model able to perform in different domains just changing some certain parameters.
- Knowledge loss: The problem of knowledge loss is highly related to scalability, the efficiency of the system is inversely proportional to the size of the knowledge base. Most common solution when system efficiency drops is merging knowledge and discarding part of this knowledge. The approach of this thesis tries to fix the problem of knowledge loss, maintaining the efficiency of the algorithm.
- Evaluation Framework: The area of user modelling lack of proper mechanisms to perform comparable evaluations. Evaluation is a non-standardized and time-consuming task. The area of user modelling has usually borrowed evaluation mechanisms from the areas of Machine Learning and Information Retrieval. One of the objectives of this thesis is to adapt those evaluation mechanisms to the area of user modelling and propose a whole evaluation framework. Thus, an evaluation methodology together with a set of metrics to specifically evaluate statistical user models will be proposed. Finally, a benchmark for user modelling is presented in order to compare and replicate evaluations.

1.3 Approach to solution

As stated in the *Objectives section* of this thesis, the research problem addressed, not only focuses on developing a new user model under the collaborative approach, but also involves addressing the problem of evaluation of statistical user models.

1.1.1 Statistical user model

On the one hand, the solution to the statistical user modelling problem will be addressed by adapting and employing an R-Tree structure Guttman, (1984) to build a new user model proposal that meets the requirements described on the *Objectives section* of this thesis. This data structure hierarchically divides the space into several overlapped sub-spaces. Therefore, the whole universe of users will be split into several sub-spaces depending on user's similarities. According to R-Tree performance, new elements are added to the leaves of the tree while the rest of the nodes (non-leaves) will store knowledge about past users. Thus, it can be stated that this approach overcomes the problem of knowledge loss since every single user is represented with different detail at each level of the tree. Starting from the root until the conjunction of all user's descriptions on the leaves of the tree.

Therefore the R-Tree structure alleviates the problem of knowledge loss. In fact the main reason to alleviate this problem is in the efficiency of this structure even when the knowledge base grows. Since the R-Tree is a structure that grows width wise even when it is traversed in lengthwise, the tree searches are very efficient even when the volume of data stored grows immensely. This advantage of the R-Tree provides a fix to the problem of the response times and the problem of scalability. Once these two points are clear the problem of knowledge loss is somehow eased since the growth of the knowledge base does not directly impact the performance of the model.

1.1.2 Evaluation framework

The area of user modelling requires its own evaluation mechanisms that can be defined departing from those used in Machine Learning and Information Retrieval areas but adapting and extending these mechanisms to the specific problem of user modelling. The key to build a complete evaluation framework is providing proper metrics and an overall methodology together with a set of experiments. All these components have to be mixed in order to create a re-usable benchmark. The evaluation methodology should answer the question, how the evaluation process should be accomplished? The set of experiments must be described incrementally departing from a baseline experiment and improving this reference trial. On the other hand, the metrics defined should describe which performance features take into account when evaluating and how to measure these features. All in all, these three components define the evaluation frameworks and enables comparable and replicable evaluations.

1.4 Document structure

The explanation of how this thesis has evolved and how it has been achieved is going to be divided in different sections. In particular this document contains six main sections: *Introduction, Theoretical part, State of the Art, Proposal, Evaluation*, *Concluding Remarks* and *Future Works*.

The first chapter explains the motivation of this thesis and the main reasons to pursuit a new approach in statistical user modelling (Section 1.1). Section 1.2 analyses main objectives of the research thesis and Section 1.3 provides an overview of the thesis proposal. Once it is clear the purpose of the thesis, the second part of the thesis explains the theoretical foundations that support the development of current proposal. This section is divided in the three main areas that support this new approach, Mass storage, R-Tree structure and Data mining. Next, the third section of this document includes references and explanations of highly related works together with main works on the area and main problems, issues and barriers on statistical user modelling.

Third chapter of the document explains the proposal addressed on this thesis. This chapter explains solution taken to solve the problems found on the previous section. In this chapter, the reader can also find the software methodology followed during the development of the proposal. The evaluation of the new proposed approach is presented on the fifth block (Section 5) where the reader can find comparisons among different models and their performance in different scenarios. Finally, a complete analysis and discussion of the thesis is provided in chapter six. This section provides an outline of the main advantages and disadvantages of the solution provided together with an explanation of main issues found during the achievement of the thesis.

2 Theoretical part: Foundations

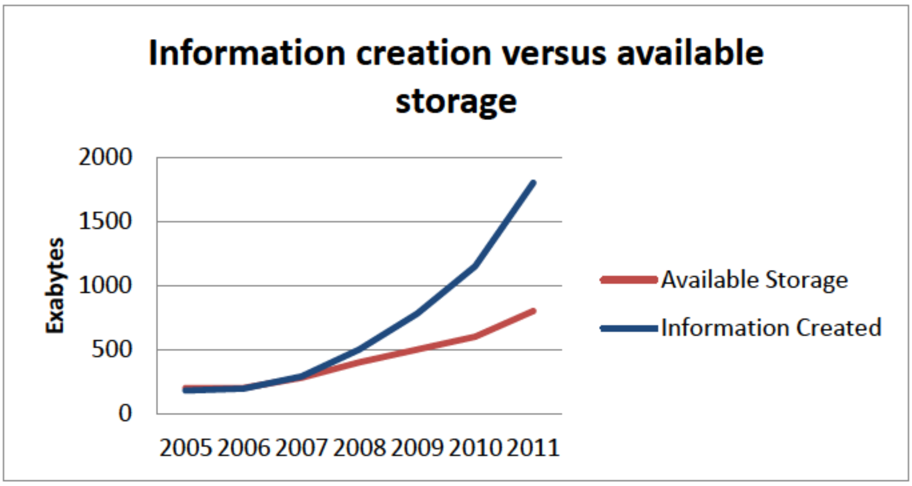
According to objectives described in Section 1.2, there is a subjacent lack of mechanisms that employ existent technologies to provide resources to achieve these goals. Some of these technologies have a large and proved background. In fact, current proposal relies on these technologies to be developed. These technologies not only enable the development of current proposal, but also provide background enough to develop alternatives and lead to new encouraging solutions to the problem of statistical recommenders under the collaborative approach. Even though, some of these tools have yet been employed to solve or alleviate main problems of statistical user models, the use of some of these technologies to this problem is completely new.

This section provides an outline of the main technologies that support this research thesis. In addition, some other technologies or foundations close to the development of this thesis will be introduced along this section. Main areas detailed in this section involves mass storage, including two different approaches, databases and file manager systems. The R-Tree structure, which is one of the main basics of this thesis; and finally an approach to data mining on the side of retrieving relevant information.

2.1 Mass Storage

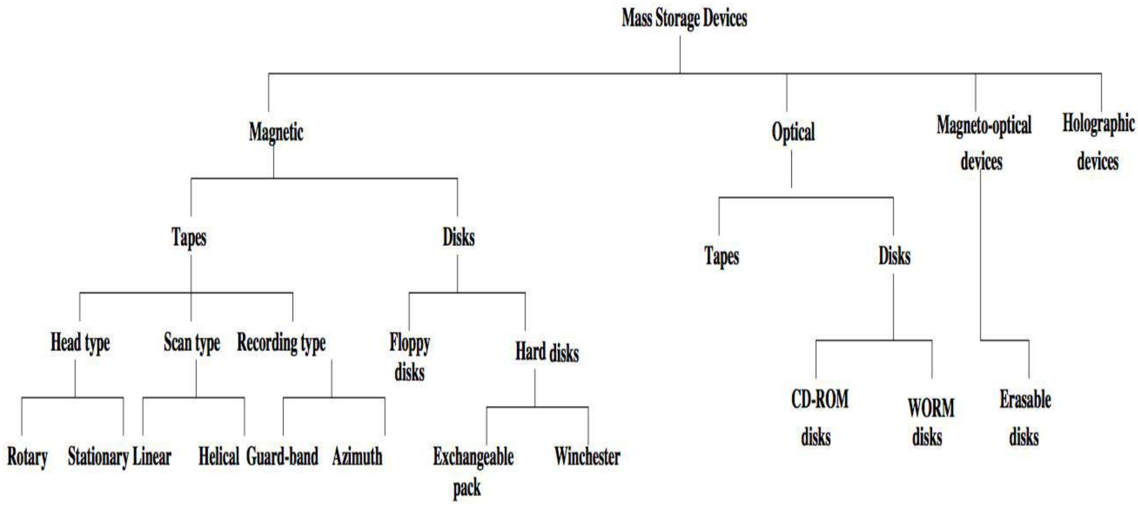
The term mass storage refers to the storage of large amounts of data in permanent devices. More specifically, mass storage refers to the lowest level of a file system that is the secondary and tertiary storage structures. Most software systems need to store data in permanent devices. Over the years the need of mass storage has existed in different areas. In fact, the rapid growth of electronic information has rapidly increased the need of mass storage. Figure 1 depicts the growing need of mass storage.

Figure 1 Reasons for Mass Storage



Some systems, like for example multimedia applications, usually need to store big amounts of data. For instance, in 1957 the highest storage density in a commercial disk was 2000 bits per square inch, in 1972 it was 800000 bits per square inch, and nowadays this density is close to 1 Tb per square inch. Thus, it can be stated that the idea of mass storage and the concept of large amounts of data depends on several circumstances such as: the time frame or the industry. In addition, there are several taxonomies of mass storage devices, main ones are depicted in Figure 2.

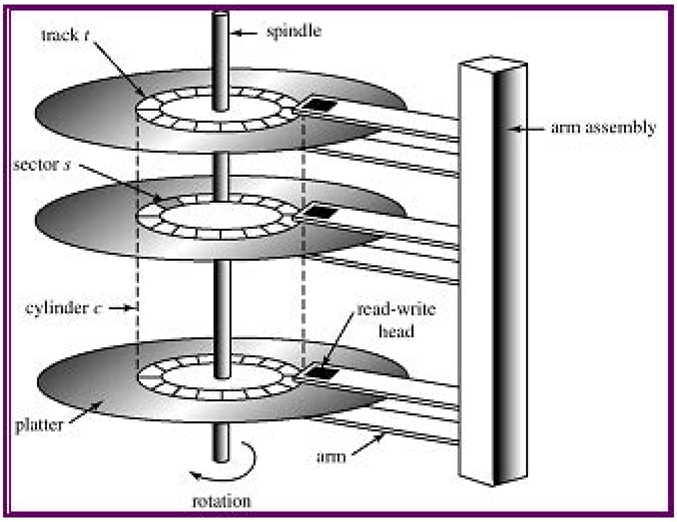
Figure 2 Taxonomy storage devices



According to Figure 1 and Figure 2, mass storage devices have evolved over the years and are still evolving. On the other hand, the capacity of mass storage devices has increased by several orders of magnitude during the last years and it is still growing. In fact, the apparition of multimedia applications unleashes the enormous growth of mass storage capacity, since this type of applications requires storing and handling big amounts of data. Mass storage is mostly restricted to devices (electronic hardware) that store information and supports some protocols for sending and retrieving information over a hardware interface. Information stored in such devices range from executable programs until documents or images Hoagland (1972).

There are different types of mass storage, being magnetic disks and magnetic tapes the most widely spread. Magnetic disks provide the majority of secondary storage for modern computers. Magnetic disks consist of several platters; these platters are covered with a magnetic material able to record information. The read/write heads attached to the disk arm are responsible for reading and writing information on the disk. The surface of every platter is divided into tracks and the tracks are subdivided into sectors. The set of tracks at one arm position forms a cylinder. Figure 3 depicts the structure of a common magnetic disk.

Figure 3 Magnetic Disks Structure

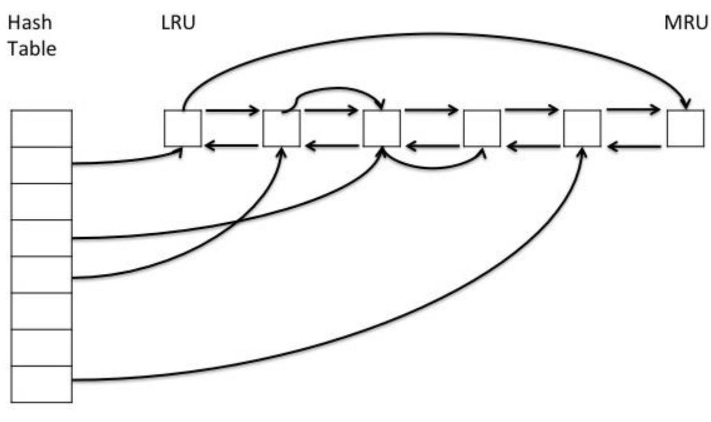


There are several parameters that define a magnetic disk, for example the speed of the disk when it is in use most disks rotate between 60 and 250 times per second. In fact, disk speed depends on two parameters: the transfer rate and the positioning time. Transfer rate is the speed at which the data flows from the disk to the computer. On the other hand, the positioning time (random access time) consists of the time that the disk arm employs to be on the desired cylinder (seek time), plus the time to rotate the desired sector to the disk head (rotational latency). Usually, disk transfers several megabytes per second and their seek times and rotational times are on the order of milliseconds.

Meanwhile, magnetic tapes were usually used as secondary storage. However, magnetic tapes have some limitations such as: slower access times (several orders slower than magnetic disks) or much more slowly random access times. Usually thousand times slower than magnetic disks. Even when magnetic tapes can store from 20 GB to 200 GB of data, they were progressively replaced by magnetic disks.

Every mass storage device such as: magnetic tapes or magnetic disks has to deal with several problems to provide good performance. These devices usually have to deal with several input/output requests that may occur at the same time. Scheduling is usually a proper solution to improve performance and enables sharing devices among processes. However, scheduling is just one way to improve performance, several other means related to the idea of using storage space in main memory or using techniques like buffering, caching and spooling can also improve performance. The principles of buffering, caching and spooling are pretty similar but the reasons to employ any of these techniques are different. Buffer usually refers to a region of physical memory storage used to momentarily store data while this data is being transferred from one place to another. Usually the data is stored in a buffer just before an input device is retrieving it or just before it is sent to an output device. There are three main reasons to use a buffer. First reason is to deal with a speed gap between the sender and the receiver. Second reason to use buffers is related to data transfers sizes, when the sender and the receiver have different data transfer sizes, a buffer is needed to avoid fragmentation and provide message reassembly. The third use of buffer is usually related to consistency reasons. This is to guarantee that when some application calls for example the system call write the content written is exactly the one at the moment of the call, no matters when the call is being executed. On the other hand, a cache is a component that transparently stores data in order to provide faster access to that data. Thus, access to cached data is usually more efficient than accessing to the original data. Main difference between a buffer and a cache is that a buffer might have the only copy of some data while a cache has a copy of the data that resides elsewhere. Even though, buffering and caching have different purposes, a region of memory can be used for both techniques. In fact, buffers are usually used as caches to improve the input/output efficiency for files shared by different applications. Another example of caching is applying caching to the web and it is known as web cache. Web cache is a mechanism for temporary storage of web documents to reduce bandwidth usage and alleviate server lags. Finally, spooling is an application of buffering for output devices that do not support interleaved data streams. Main example of spooling devices are printers. Printers can only attend one request at the same time, several printers may wish to print the same work concurrently. To solve this issue, the operating system spools the output to a separate disk file for every printer. These techniques usually employ algorithms to manage the cache memory. These algorithms usually try to provide blocks accessing the cache only. In case the block is not in the cache, that block will be read in the cache. Thus, next queries over the same block do not need accessing the disk again. Thousands of blocks are in the cache and it is needed to decide quickly if a block is present or not. Most common solution involves hashing the device and disk addresses and figure out the result in a hash table. All blocks with the same hash value are put together in a linked list. It might happen that a new block has to be inserted in a full cache; in that case, some blocks must be removed. Several algorithms can be used to decide which block should be removed; main strategies are FIFO, LIFO, LRU and MRU. Figure 4 depicts buffering scenario having the hash table and the collisions chain for all block having the same hash value.

Figure 4 Buffering structure



Even thought, this scenario enables accessing blocks quickly, some algorithms such as LRU might present some disadvantages and may become impracticable. If some important block e.g. an i-node is read into the cache and modified, if the system crashes before this block is re-written, the file system will be inconsistent. Therefore, some considerations must be taken into account before applying any algorithm for block replacing. First consideration involves considering if the block is going to be needed again and second one consider whether the block is essential for the consistency of the file system. To alleviate these problems, blocks are divided into categories such as i-node block, indirect blocks, directory blocks and full data blocks. Thus, blocks that are supposed to be needed again will go to the end of the queue. In addition, essential blocks to the file system consistency must be written immediately if they have been modified. Re-writing these blocks immediately eliminates the possibility of having some inconsistencies.

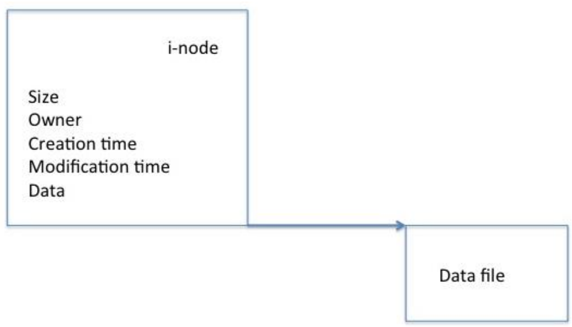
2.1.1 File systems

As has been explained before, computers can store information on various storage media, for example magnetic disks, optical disks, magnetic tapes, etc. The information stored on these media can be accessed by the operating system. The operating system provides a logical view abstracting from the physical details of the storage devices. This logical storage unit is the file. The content stored on files is persistent and is mapped to physical devices for the operative system. Therefore, files contain related information and are stored on secondary storage. A file can contain many different types of information, such as: source programs, object programs, executable programs, text, images or sounds. Depending on the content, the file has a certain structure, for example a text file is a sequence of characters. In general, all files have some attributes that define the file and its content. Though some attributes can vary depending on the operating system, most common attributes are the followings:

- Name: The name is used to refer to the file and is the only human readable information.
- Identifier: Identifies the file within the file system. The identifier is numeric information.
- Type: Information required for those systems that support different types of files.
- Location: A pointer to the physical device and to the location of the file on that device.
- Size: Current size of the file and maximum allowed size for this file.
- Protection: Determines who can read write and execute the file.

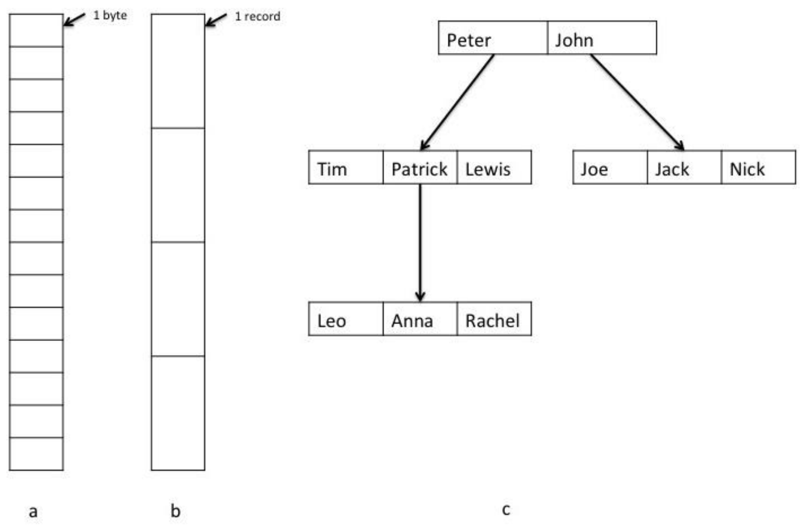
The information about the files is stored in the directory structure. Unix based system use a structure called i-node to keep all this information. Figure 5 depicts the i-node structure for a data file. The i-node keeps all the information about the file except from the name.

Figure 5 I-node information



As have been explained before, there are different types of files; Figure 6 summarizes main categories of files depending on the structure.

Figure 6 File Types



The file (a) is an unstructured file that consists of a sequence of bytes. In fact, the operative system just see bytes inside this type of files (Unix and Windows use this approach). In addition, this approach is the most flexible since programs and the operative system can store almost anything in files. Second approach, or structured file, is depicted in file (b), which consists of a sequence of fixed-length records. Each record might have a different structure. Third approach is displayed in file (c) and it is the tree structure. Each node of the tree contains records of different structure and length, each record has a key field. The tree is sorted using the key field to enable fast searches over the file. On the other hand, most operative systems support several types of files, for instance UNIX and Windows support regular files. Regular files used to be ASCII files or binary files. Regarding file access, early operative systems provided only sequential access to files. Sequential access did not enable skipping bytes of the file. Lately operative systems provide random access to files, which is essential for many applications as database systems. Main issue with random access is determining the starting point to read, two different approaches were developed. First approach states that the reading operation must determine the starting point. Second approach provides the functionality seek that set the starting position in the file.

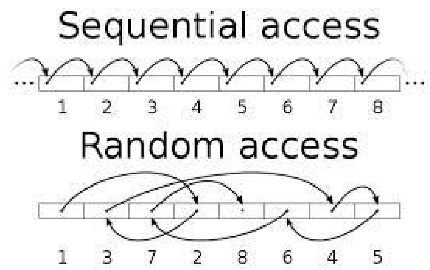
Operative systems must provide several functionalities to store information in files and retrieve such information later. Most common operations are:

- <u>Create</u>**:** Create an empty new file and sets its attributes.
- <u>Delete</u>**:** Delete the file permanently from the disk.
- <u>Open</u>**:** The open call moves partially the file to main memory to be used for some application.
- <u>Close</u>**:** Close the file and frees up internal table space.
- <u>Read</u>**:** Enable reading data from the file. Usually the caller must specify how much data is needed and a buffer to store this data.

- Write: Write data in the file at current position. If current position is the end of the file, the size of the file increases. If current position is in the middle of the file, data is overwritten.
- Append: Add data to the end of the file.
- Seek: Seek is used to set the starting position for random access files.
- Get attributes: Return the attributes of the file.
- Set attributes: Enable modifying some of the attributes of the file.
- Rename: Change the name of the file.

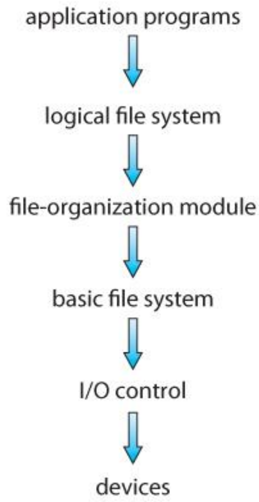
All these operations enable handling files, storing information and reading information. However, accessing files can be done in different ways. Most common access method is sequential access where the information in the file is accessed in order, one record after the other. Some operating systems provide functionalities to reset the pointer to the beginning of the file or even skip forward a number of records. On the other hand, direct access files contain fixed length logical records and allow programs to rapidly access records in no particular order. Figure 7 depicts the two access cases.

Figure 7 Sequential Access versus Direct Access



The direct access method relies on a disk model of a file where the disk provides random access to any block. Direct access files are very efficient for immediate access, usually databases supports this random access. Despite all these operations provided, some devices are more suitable for secondary storage than others. For instance, magnetic disks provide several features that make this device the best secondary structure to store data. Magnetic disk can be rewritten, that is a block can be read, modified and written back. A disk provides direct access to any block. In terms of efficiency, the input output operations rather than transferring one byte, transfer units called blocks. Thus, blocks are the minimum transfer unit, blocks sizes range from 32 bytes to 4096 bytes. The operative system enforces one or more file systems that are responsible for providing efficient access to the disk. The design of a file system usually copes with two main problems, how the file system looks to the user and which algorithms and data structures guarantee a mapping between the logical side and the physical secondary storage. The file system imposed by the operating system is usually composed of different layers. Each layer uses the features of the lower level and provides features to the higher level. Figure 8 provides an overview of a common layered file system.

Figure 8 Layered File System



The lowest layer, the input output control, contains the device drivers and transfer information between main memory and the disks. Thus, the basic file system layer only needs basic commands to read and write data to or from the disk. The file organization module knows about files and their logical and physical structure. Therefore, this layer can map and translate logical block addresses to physical block addresses. This layer is also responsible for handling empty blocks. The logical file system manages metadata information containing all the file system structure except the data itself. In fact, this layer is in charge of the security of the file system, of the file organization module and of the file structure.

2.1.2 Databases

Since the beginning of computation, storing and manipulating data have been a major focus. The history of databases dates from the 1960 when Charles Bachman designed the Integrated Data Store. Since 1960 until now the amount of electronic data available has been constantly growing and the need to organize this information is becoming each time more important to developers and systems. To handle such amounts of information, users need tools that simplify managing data and extract relevant information from data. Databases provide functionalities to store data, to access data and to retrieve information in a short period of time. Several definitions of databases can be found in the literature, one of the most widely known is the following Ramakrishnan and Gehrke, (2000). A database is a collection of data describing the activities of one or more related organizations. An example might be a database for a university containing entities such as: students, faculty or courses. Relationships between entities for instance students enrol in courses.

The aggregate of data, hardware and software responsible for the management of the database is called the Database Management System (DBMS). The main function of the DBMS is to provide efficient methods to access the data. These mechanisms should be not only efficient but also reliable mechanisms. In fact, most DBMSs have to deal with several users trying to access data simultaneously. Elmasri and Navathe, (2006) provided another definition of databases. They stated that a database is a collection of related data. Understanding data as facts that can be recorded and that have implicit meaning. The need and use of databases and DBMSs is growing rapidly. In addition, all databases have some inherent properties:

- A database represents aspects of real world (Universe of discourse). Changes in the universe of discourse are reflected in the database.
- A random collection of data without implicit meaning cannot be referred as database.
- A database is designed, built and used for some specific purpose. It has some users and applications with specific needs.

In addition, databases and DBMS provides some advantages to users, main benefits are the following.

- <u>Improved availability</u>**:** Same information is available to different users.
- <u>Minimized redundancy</u>**:** Databases are designed to reduce redundancy. It means that the cost of storage will be also reduced.
- <u>Accuracy</u>**:** Accurate and up-to-date data is a property of integrity. In addition, the chances of making a mistake are higher if you have to make changes in several places.
- <u>Improved security</u>**:** Multiple users can access resources. Even though, this is a reason for risk, DBMS manage security even in these scenarios. DBMS enforce integrity constraints.
- <u>Data independence</u>**:** Applications should not be aware of details of data representation and storage.
- <u>Efficient data access</u>**:** DBMS are in charge of retrieving data efficiently and uses several techniques to provide this feature.
- <u>Data administration</u>**:** When several users share data, centralized administration provides several improvements. For example redundancy reduction.
- <u>Crash recovery</u>**:** DBMS protects users from the effects of system failures. For example providing backup and recovery functionalities.
- <u>Development time cost</u>**:** DBMS supports several functionalities that are common to applications accessing data.
- <u>Java embedded code</u>**:** Some DBMSs provide functionalities to embed Java source code in its databases. This functionality improves the performance of the system.
- <u>Buffering</u>**:** Some DBMSs provide the ability to maintain in cache memory the part of the database that is being accessed at each time.
- <u>Improved performance</u>**:** BDMS might provide specific functionalities to improve the performance of some systems. For example, modifying the block size may be a performance improvement for some specific applications.
- <u>Performance variables</u>**:** Some DBMSs enable modifying variables like PCTFREE, PCTUSED, which respectively modify the empty space in a block and the free space needed to allow new insertions in the block.

Both file systems and databases provide proper features for mass storage and for dealing with unstructured data. However, the lack of proper information retrieval mechanisms leads to inaccessible data or inefficient access to data. The need of information retrieval is satisfied with the introduction of Databases and File Systems that provide mechanisms to access data. Both

Databases and File Systems have some advantages and disadvantages that make one mechanism more suitable for certain purposes than the other. For example, Databases are more adequate for certain processes such as: complex queries or concurrent requests. On the other hand, the performance of Databases cannot be enough on real-time applications. For those systems requiring real-time responses, a file system using caching mechanisms is more efficient than databases.

2.2 Multi-key access

In order to speed up the access to indexing data becomes crucial. An index is a data structure that improves the speed of data retrieval operations. Though, indexes speed up retrieval operations, maintaining indexes have a cost of additional storage to keep a copy of the data. However, index size is usually much smaller than the size of the original data. Therefore, indexes provide a quick access to data without having to search every single record. There are several categories of indexes, primary and secondary indexes, dense and sparse indexes. Indexes are a common solution to single-key access to data where the data is searched for a single-key and indexes provide quicker access than searching the overall records. However, as data becomes more and more complex, queries have evolved to turn from single-key access to multi-key access. Most common index solutions are not adequate for multi-key access, leading to serial access to data when multi-key access is required. Nonetheless, there are some structures suitable for multi-key access. For example, Grid files, Bitmap indexes and R-Trees provide efficient access when performing multiple key accesses to data. Though this research thesis is highly related with the R-Tree, the explanation of this structure will be skipped on this sub-chapter, however a fully detailed explanation of the R-Tree can be found in the following chapter.

Therefore, the rest of this section focuses on the grid files and the bitmap indexes as solutions to multiple key accesses. As explained, a wide selection of file structures is available for: accessing collections identified by a single key; sequentially allocated files; tree-structured files and even hash files. These structures allow the execution of common operations such as: find, insert or delete when searching by a single query. The history of multi-key access can be traced back to the history of single-key access. In fact, structures developed for multi-key access have their origins in structures designed for single-key access. Though this structures address the problem of multi-key access, most of them have deficiencies when dealing with highly dynamic environments.

A grid file can be considered as an access method that splits a space into grids where one or more cells of the grid refer to a small set of points. The grid file is thought to speed up multi-key queries involving one or more operators. The grid file contains a single grid array and one linear scale for every search attribute. The number of dimensions of the grid is the number of key attributes. In fact, multiple cells can point to the same bucket. Therefore, searches involve: locating the row, locating the column and following the pointer. On the other hand, bitmap indexes are also designed for multi-key access, but its structure is simple like an array of bits. Records in a relation should be easy to retrieve. Bitmaps are especially useful when the number of distinct values is small. Though these two structures provide a solution to the problem of multiple key access, all searching techniques fall in some aspects. The R-Tree is an encouraging approach to efficiently face the problem of multi-key access.

2.2.1 R-Tree

The Rectangle-Tree (R-Tree) was introduced in the middle of the eighties with the idea of alleviate the problem of handling spatial data efficiently Guttman, (1984). Spatial data usually covers multi-dimensional areas, which prevents the use of classical indexing structures. Classical indexing structures, e.g. B-Trees Comer, (1979), are non-efficient for multi-dimensional problems. The R-tree represents objects by intervals in several dimensions. In fact R-Tree has been proved to represent spatial data in more than two dimensions.

2.2.1.1 R-Tree structure

An R-Tree is a height-balanced tree where the nodes contain pointers to data objects. Main advantage of R-Tree is that a spatial search only requires visiting a few nodes. The structure of the R-Tree comprises the spatial data by Minimal Bounding Rectangles (MBR), Papadias and Theodoridis, (1997). Each MBR is comprised again by another MBR continuing this structure until the root of the tree that comprises all objects over the R-Tree. In addition, the index of the R-Tree is completely dynamic in the sense that inserts and deletes can be mixed with searches and no periodic reorganization of the index (tree) is required. Figure 9 and Figure 10 depict main structure of an R-Tree.

Figure 9 R-Tree MBR representation

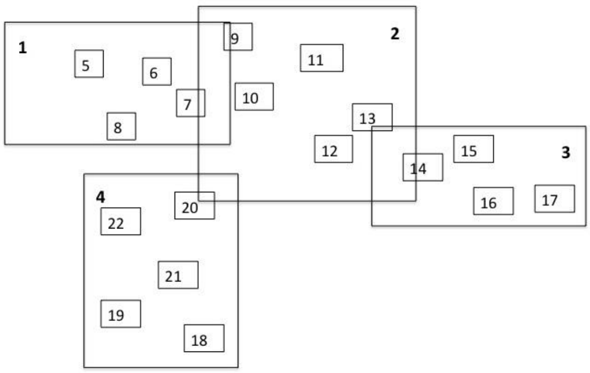
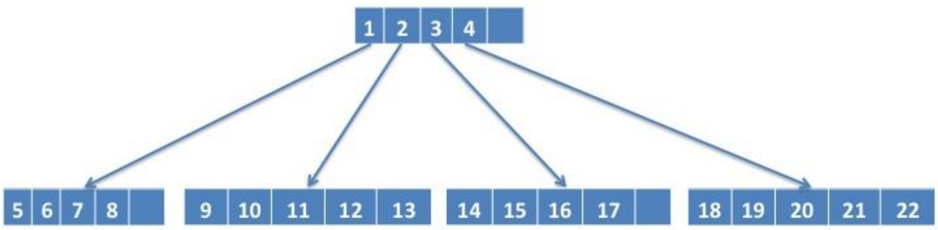


Figure 10 R-Tree



According to previous figures, the elements 1,2,3,4 are in the root node, for example the element 1 comprises the elements 5,6,7,8 with the Minimal Bounding Rectangle. There are some properties that every R-Tree satisfies:

- Every leaf node contains between m and M index records except from the root node.
- Each leaf node is represented by the smallest rectangle that spatially contains the whole n-dimensional data in the node.
- Every leaf node except from the root contains between m and M children.
- The root node contains at least two children unless it is a leaf.
- The tree is height-balanced. All leaves are in the same level.

R-Tree has two different typologies of nodes, the leaf-node and the non-leaf node. Spatial databases or multi-dimensional information consists of tuples representing spatial objects. Each tuple has a unique identifier that enables retrieving the tuple. Leaf-nodes contain index entries of the form (I, tuple-identifier) where tuple-identifier is the tuple in the database and I is an n-dimensional rectangle containing the spatial objects. In addition, I can be seen as a set of intervals I = (I0, I1, I2, …., In-1) where n is the number of dimension and Ii represents the extent of the object along dimension i. On the other hand, leaf-nodes contain entries of the form (I, child-pointer) where child-pointer is the address of the child node and I covers all elements in child node.

Figure 11 Root element 1 covers all elements in child node

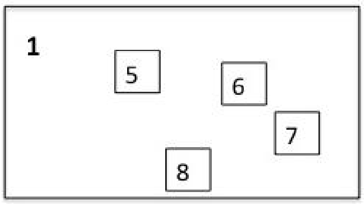
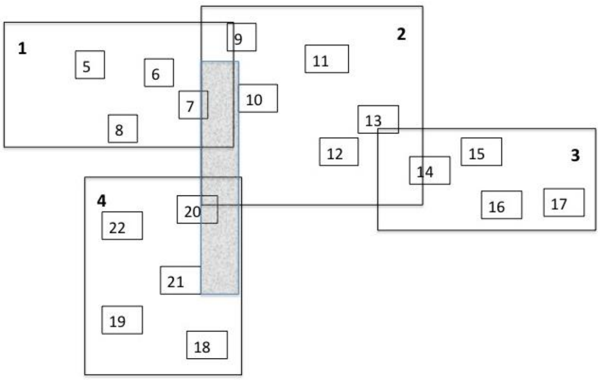


Figure 11 can be compared with Figure 10 to understand how the root element 1 covers all elements in its child node with the smallest rectangle. Variable m represents the minimum number of elements that every node except from the root node might have. On the other hand, M is the maximum number of elements in a node. Therefore, the minimum number of entries in a node depends on M, having $\frac{M}{2} \geq m$. In addition, having that N is the number of index records, the maximum number of nodes in the tree is computed as: $\frac{N}{m} + \frac{N}{m^2} + 1$. Notice that the parameter m will be responsible for the height of the tree and for the performance of the algorithms over the structure.

2.2.1.2 Algorithms

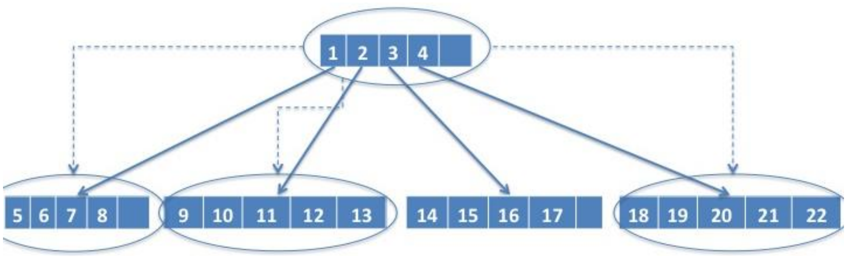
R-Trees used to rely on four main algorithms to perform. Those algorithms are: search, insert, delete and split. The search algorithm for R-Tree starts in the root node and descends to leave nodes similarly to the search in a B-Tree. However, in the case of the R-Tree, more than one sub-tree under current node might be searched. That happens due to the multidimensional properties of information and because of the overlapped regions in spatial information (See Figure 10). Figure 12 shows how overlapped regions might be needed to be explored during one search.

Figure 12 Search overlapped regions R-Tree



The grey rectangle in Figure 12 shows the area that is necessary to explore to perform a search in the R-Tree. Figure 13 shows the path required for that search.

Figure 13 R-Tree search path



The search algorithm pseudo code is the following: (1) having the root node R; (2) find all index records whose rectangles overlapped the search rectangle - see Figure 10.

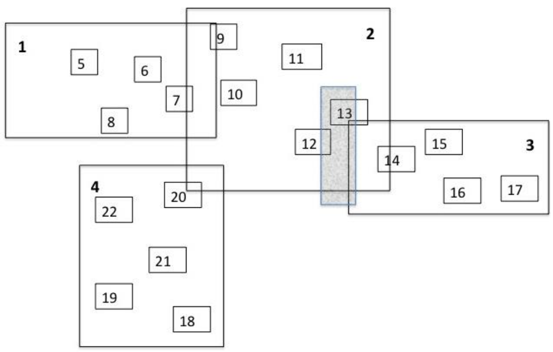
Step	Description
1.- Search sub-tree	If R is not leaf then Check entry Ei to see if its rectangle overlaps S FORALL overlapped entries invoke Search with root node pointed by Ei
2.- Check Relevance	If R is a leaf check all entries E to see whether Ei overlaps S If so E is a relevant record

The insertion process is quite similar to inserting in a B-Tree, new records are always inserted on the leaves nodes of the tree. Leaves nodes can overflow and split; and there might be split propagation up the tree. The pseudo code to insert new records in an R-Tree is detailed next. Notice that this pseudo code requires some additional methods.

Step	Description
1.- Insert	Find position for new entry E; invoke ChooseLeaf to find a leaf L to place E.
2.- Add record to leaf	If L has room for a new entry then add E. Else invoke SplitNode to obtain L and L' containing E and all entries in L.
3.- Propagate changes	Invoke adjust tree on L and on L' if a split was required.
4.- Grow tree	If split propagation causes the root to split, then create a new root whose children are the two resulting nodes.

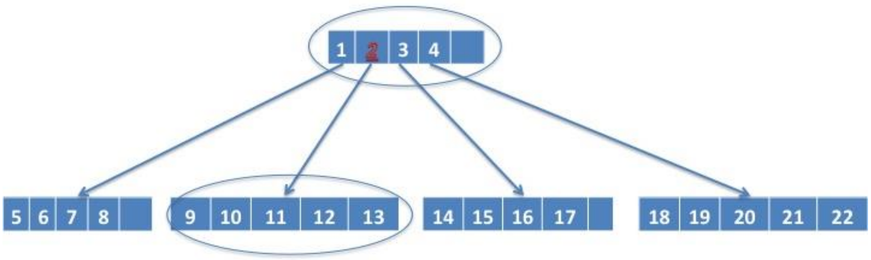
Figure 14 shows how a new insertion is done in the tree of Figure 10, showing the MBR representation.

Figure 14 R-Tree insertion



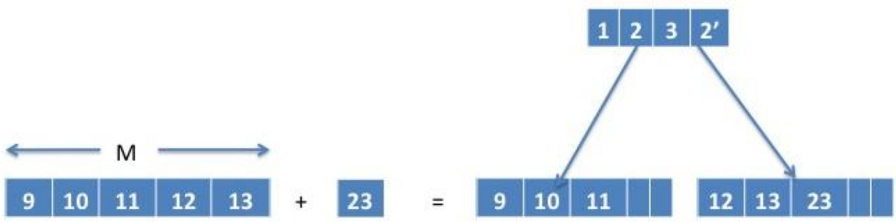
In Figure 14, the grey rectangle is inserted in the R-tree. To select the best node to insert, the method ChooseLeaf is invoked. Figure 15 depicts how this method chooses the best way to insert the new record.

Figure 15 R-Tree insertion path



First of all, the algorithm explores the root node to find the most similar record. In this case, the most similar node is 2. After that, the child node is explored and the record gets inserted since it is a leaf node. Notice that the ChooseLeaf method has to select between the node 2 and 3 to insert the new record. The selection goes for the node 2 because its rectangle needs less enlargement than the rectangle of the node 3, to fit the new record. However, in this particular case, the leaf node does not have enough room to place the new record, thus the node produces an overflow and invokes the split procedure. Figure 16 depicts this scenario.

Figure 16 R-Tree overflow and split

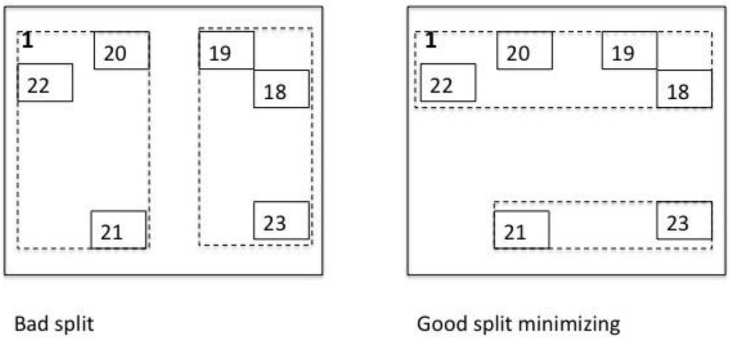


The split procedure tries to minimize rectangles, according to Figure 16, records (9, 10, 11) can be covered by the minimum rectangle and elements (12,13,23) can also be covered by a minimum rectangle. Therefore the elements are distributed according to Figure 16. On the other hand, the delete process differs from the deletion in a B-Tree. The pseudo code to delete the index record E from the R-Tree is the following.

Step	Description
1.- Find node	Invoke FindLeaf to search the node L that contains E. If record is not found stop
2.- Delete	Remove E from L
3.- Propagate changes	Invoke CondenseTree on L
4.- Shorten Tree	If the root node has only one child, the child becomes root node

In addition, an explanation of the CondenseTree method will be provided. This method makes the difference between R-Tree deletion and B-Tree deletion. In an R-Tree, if a node has an underflow, it is deleted and inserted again. On the other hand, in the B-Tree the node would be fused with another node. In addition, the deletion process makes R-Trees more efficient than B-Trees since the insertion process can be used to develop the deletion. Furthermore, after each deletion and re-insertion, the spatial structure of the tree is improved. The insert and delete procedures use the split mechanism to save the tree structure. The split procedure gets activated when trying to add a new entry to a full node. That is a node containing M entries. In addition, the split procedure is responsible for guaranteeing that the probability of both nodes being explored in following searches is as lowers as possible. As previously mentioned, the split procedure must minimize the total area of the resultant rectangles. Figure 17 shows a bad split and a good split taking into consideration the areas of the resultant rectangles.

Figure 17 Split procedure R-Tree



As can be seen in the previous figure, the second split minimizes the area of resulting rectangles and it is considered better than the first split.

2.2.1.3 R-Tree performance

Several performance test where done with an R-Tree trying to find the best performance and the best values for both M and m Guttman, (1984). The test was done using two-dimensional data and having the minimum number of entries in a node $\frac{M}{2}$, $\frac{M}{3}$ and 2. In addition, five page sizes were used during the test. Table 1 depicts main issues of pages' sizes.

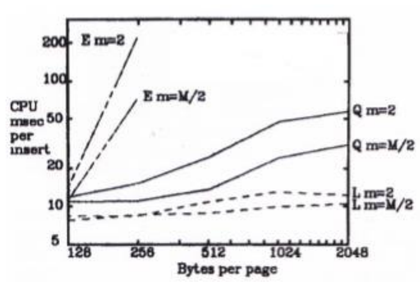
Table 1 Test issues

Bytes per page	M
126	6
256	12
512	25
1024	50
2048	102

Each test performed starts reading data and inserting in an empty tree. After inserting, the test continues searching randomly generated rectangles; and finally, deleting some rectangles. In addition, several algorithms were compared to perform the splitting process. These algorithms are compared during the test and can be identified according to the following notation, E (Exhaustive algorithm), Q (Quadratic algorithm) and L (Linear algorithm).

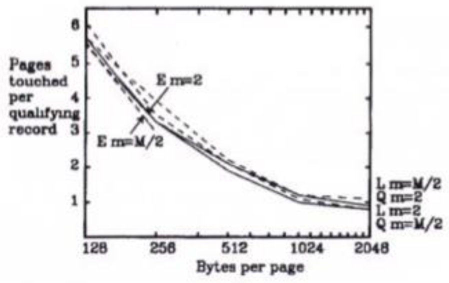
First test involves inserting records in an R-Tree. Figure 18 depicts CPU cost for inserting the last 10% records of each proposed experiment with different values of M and m.

Figure 18 CPU cost insert records



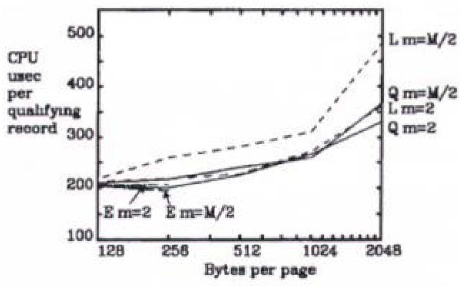
As can be seen in the previous figure, the best algorithm in performance is the linear algorithm while the exhaustive algorithm needs more time even with fewer pages. Notice that the linear algorithm keeps the CPU cost even when the number of pages increases. Second test checks the performance of the structure when searching records. This test invokes the function search 100 times. Two different measures were taken: first one involves count the number of pages visited per search; and the other measures the CPU cost. Figure 19 displays performance test when counting the number of pages per record searched.

Figure 19 Search pages visited



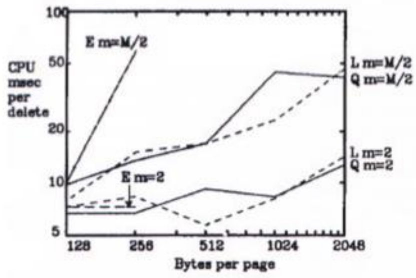
Notice that even three algorithms, which behave similarly (search does not require splitting), the exhaustive method behaves slightly better since its index is a little better. Figure 20 shows CPU cost when performing the searches.

Figure 20 Search CPU cost



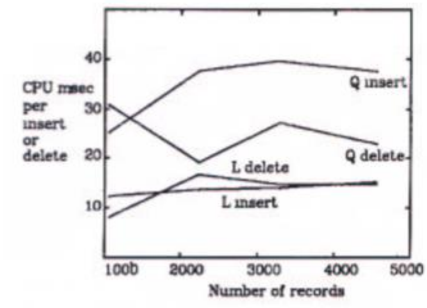
Once again, the exhaustive algorithms provide better performance because of its index. In addition, a deleting test was performed. One index record for every tenth data item was removed. Figure 21 depicts the CPU costs for each algorithm when performing the deletions.

Figure 21 Delete CPU cost



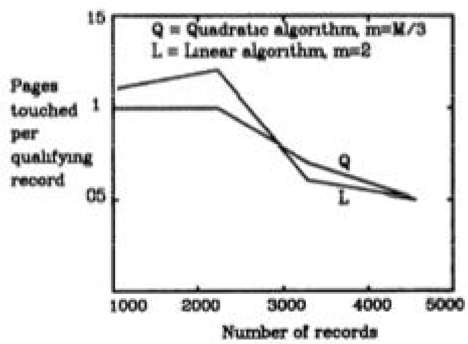
The algorithms, that perform better, were the linear method and the quadratic method. However, the results were affected by the value m, lower values of m yields to higher rates of underflows and re-insertions (might cause overflows). Guttman presented some additional tests for the R-Tree. For these new experiments, the same operations were performed but using samples containing 1057, 2238, 3295 and 4559 rectangles. Main purpose of these tests is measuring the R-Tree performance depending on the amount of data in the index. Main parameterizations of these experiments are: Linear algorithm with $\mathrm{m} = 2$ and quadratic algorithm with $\mathrm{m} = \frac{\mathrm{M}}{3}$ page size is 1024 bytes and $\mathrm{M} = 50$. Figure 22 shows the results of the text to compare how insertions and deletions may be affected by tree size.

Figure 22 Insert and delete CPU cost



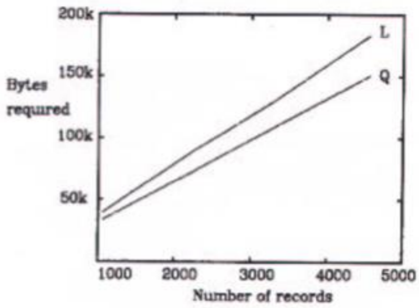
Guttman noticed the test containing 1057 rectangles produced a tree with two levels while the other tests yield a tree with three levels. It can be seen that quadratic algorithms remain almost constant for inserting, except from the moment where the tree grows in height. In addition, the linear algorithm remains almost constant during all the insertions. Guttman also noticed that no split occured during the deletion because of the small number of items. The small jump that happens during the delete process is due to the new level of the tree. Figure 23 confirms that both configurations have almost the same performance even during the search tests.

Figure 23 Search versus amount of data



Finally, Figure 24 confirms that almost every space in the R-Tree is used for leaf nodes.

Figure 24 Space required by an R-Tree



According to Figure 24, it can be concluded that the number of leaves in the tree increases linearly with the amount of data in the index. Finally, according to Guttman, the R-Tree has been proved to be useful for indexing spatial objects. In addition, the linear algorithm has been proved to behave as good as more expensive algorithms even performing faster than these algorithms.

2.2.1.4 R-Tree alternatives

Though, R-Tree have played an important role in terms of indexing multidimensional data, there are several alternatives that have to be addressed. R+-Trees were proposed as a structure that avoids visiting multiple paths during point location queries. And thus, the retrieval performance could be improved Timos, et al., (1987). Moreover, MBR overlapping of internal modes is avoided. This is achieved by using the clipping technique. In simple words, R+-Trees do not allow overlapping of MBRs at the same tree level. In turn, to achieve this, inserted objects have to be divided in two or more MBRs, which means that a specific object's entries may be duplicated and redundantly stored in several nodes. The search algorithm is similar to the one used for R-trees. The only difference is that duplicate elimination is necessary to avoid reporting an object more than once. However, insertion, deletion, and node splitting algorithms are different due to the clipping technique applied. In order to insert a new entry E, the insertion algorithm starts from the root and determines the MBRs that intersect. Then, it is clipped and the procedure is recursively applied for the corresponding subtrees. The fact that multiple copies of an object's MBR may be stored in several leaf nodes has a direct impact on the deletion algorithm. All copies of an object's MBR must be removed from the corresponding leaf nodes.

Time after the apparition of R+-Trees a new evolution called R*-Trees were proposed Beckmann, et al., (1990). R*-Trees are still very well received and widely accepted in the literature as a prevailing performance structure that is often used as a basis for performance comparisons. As previously explained R-Tee relies on the area minimization of each MBR. However, R*-Trees examines other alternatives. The main alternative is minimization of the area covered by each MBR. This criterion aims at minimizing the dead space, to reduce the number of paths pursued during query processing. This is the single criterion that is also examined by the R-tree. Second criterion is minimization of the overlapping between MBRs. Since the larger the overlapping, the larger is the expected number of paths followed for a query, this criterion has the same objective as the previous one. Finally, maximization of storage utilization to involves fewer nodes in query processing.

Apart from the aforementioned R-tree variations, a number of interesting ex- tensions and adaptations have been proposed that in some way deviate from the original idea of R-trees. The Sphere-tree by Oosterom, (1999) uses minimum-bounding spheres instead of MBRs, whereas the Cell-tree by Guenther (1989) uses minimum bounding polygons designed to accommodate arbitrary shape objects. Similarly to Cell-trees, Jagadish (1990) proposed independently the structure of Polyhedral trees or P-trees, which use minimum bounding polygons instead of MBRs. The QR-tree proposed in is a hybrid access method composed of a Quadtree and a forest of R-trees Fu, et al., (2003). The Quadtree is used to decompose the data space into quadrants. An R-tree is associated with each quadrant, and it is used to index the objects' MBRs that intersect the corresponding quadrant. Performance evaluation results show that for several cases the method performs better than the R-tree. However, large objects intersect many quadrants and therefore the storage requirements may become high due to replication. Finally, the S-Tree relaxes the rule that the R-tree is a balanced structure and may have leaves at different tree levels Aggarwal, (1997). However, S-trees are static structures in the sense that they demand the data to be known in advance.

2.2.2 Other structures

Despite R-Trees are the core of this multi-key access structures, some other structures have to be mentioned due to their importance. For instance, Excell Tamminen, (1982), which becomes a generalization of extendible hashing to higher dimension. It applies a one-dimensional hashing and makes it extensible through the rest of dimensions. Applying a similar strategy than the one applied in grid files. Grid files are one of the simplest data structures that often outperform single-dimension indexes for queries involving multidimensional data. Grid files partitioned the space of points in a grid. In each dimension gridlines partition the space into stripes. Those points that fall into a grid line are considered to belong to the stripe for which that grid line is the lower boundary. The amount of grid lines in different dimensions may vary. Each of the regions of a partitioned space can be seen as a bucket of a hash table. Every point in that region has its record placed in a block belonging to that bucket. Overflow blocks can be used to increase the size of a bucket. Instead of a one-dimensional array of buckets, as is found in conventional hash tables, the grid file uses an array whose number of dimensions is the same as for the data file. To locate the proper bucket for a point, it is needed to know the list of values of the grid lines for each dimension. Once the bucket is decided, the point can be stored in that bucket. If there is no room for the point, two main solutions can be adopted. First one involves adding overflow blocks to the buckets and second one involves reorganizing the structure adding or moving the grid lines.

Accessing buckets in a grid file is not always a simple task, while ideally a grid file might have just few dimensions and few stripes per dimension, usually grid files have lots of dimension and stripes on every dimension. In that case, indexing dimensions used to help in terms of performance when looking for a cube. Grid files measures times to perform a file operation in terms of disk accesses.

Another remarkable structure well-known for its performance when dealing with multidimensional data is the general grid. The general grid is a flexible generic structure for storing grid information. It is usually employed to store data of arbitrary dimensions and it is also capable of dealing with unstructured grids. The grid is represented by a list of grid points along with other attributes representing each location.

Finally, the Bang file Freeston (1987) is a particularly relevant attempt of balancing the load in a regular partition pattern. It uses to work over a regular grid to guarantee linear growths. Unlike the grid file the Bang file splits a cell into two resulting subspaces differing as little as possible. This is done under the restriction that the smaller subspace is created by recursively cutting the subspace to be divided. Those cuts run cyclically through all dimensions. Usually Bang files have fewer cells than the grid files. However cells in Bang files use to be more complex than cells in grid files. Despite their complexity, these cells can be stored in a highly compressed form. They are created with a strict mechanism that enables storing cells as a pair of indexes. Bang files are organized following a tree structure.

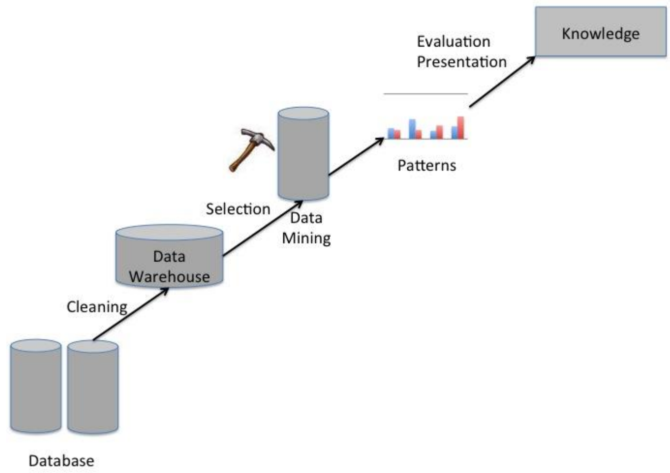
2.3 Data mining

Data mining is the process of finding (mining) trends or patterns (knowledge) in large data sets. People have been looking for patterns since the beginning of human life. Data mining area is pretty close to some areas like statistics, data analysis, knowledge discovery and machine learning. Even though, main ideas of most related areas can be employed in data mining, main difference refers to scalability, data mining tools are specially designed for large data sets. Even though, data mining provides tools to find patterns in large data sets, data mining in real world is even more. Data usually is incomplete and plenty of noise, if these issues are not corrected, some interesting patterns might be missed. Data mining can be positioned within the knowledge discovery process. This process can be divided into four steps:

- <u>Data cleaning</u>**:** Data is standardized, outliers and noise is removed.
- <u>Data selection</u>**:** Involves identified the data subset of interest and main attributes to consider, departing from the raw data set.
- <u>Data mining</u>**:** Apply data mining tools to figure out interesting patterns in data.
- <u>Evaluation</u>**:** Patterns are presented to final users

Figure 25 depicts this process starting from large data sets stored probably in a database, the cleaning and selection phases prepares the data to be treated with some data mining algorithm in order to extract those patters that leads to non-trivial knowledge in the data set.

Figure 25 Knowledge discovery process



As can be seen in previous figure first steps involving data cleaning and data selection are forms of data pre-processing necessary for the data mining process. Patterns discovered might be stored as new knowledge in the knowledge base. Thus, data mining is just one step within the knowledge discovery process, but it is the most critical step. In the process of data mining several components are needed to help the data mining engine in retrieving useful patterns. Main components are the database, the data warehouse, the knowledge base and the pattern evaluation component.

2.3.1 Data mining functionalities

This section provides an overview of main types of patterns that can be found applying data mining techniques. Data mining task can be classified into two different categories, descriptive and predictive. Descriptive tasks characterize general properties of the data. On the other hand, predictive tasks perform inferences on data in order to provide non-trivial predictions. Data can usually be associated with some classes or concepts. These classes or concepts can be obtained using data characterization, data discrimination and mixing both techniques. Data characterization involves summarizing main features of target class. Data discrimination involves comparing the target class with all other classes. There exist several methods from data summarization for example data warehouse, data summarization based on statistical methods and the data cube OLAP operation. In addition, there are some patterns that are common to most data sets and occur frequently in data. For example customers tend to purchase a mobile phone followed by a memory card. Several types of common patterns can be identified, common items, common sub-sequences and frequent sub-structures. Common items refer to several items that usually appear together. A sub-sequence contains a pattern related to several user behaviours that occur together. Finally a sub-structure can refer to different structural forms like for example trees representing a user behaviour.

Among data mining functionalities, classification or prediction involves dividing items in classes in order to predict the class of a new item. That is building a model able to predict the class of an unknown item. This model is usually built departing from the analysis of a training data set. The derived model might be presented as a set of rules, a decision tree, a mathematical formulation, a neural network and so on. Classification and prediction might need some relevance analysis to identify main attributes and help in the prediction process. Classification methods often have to deal with the problem of outliers. Most data points in a dataset can be represented with a model, however some data points does not comply with the general model, this data points are usually called outliers. Generally, data mining algorithms and tools discard outliers considering such data points as noise. On the other hand, some applications need outliers to mining interesting patterns. For example a fraud detection application must consider outliers to produce reliable patterns. Outlier's analysis is usually referred as outlier mining. Two main methods have been applied to identify outliers in a data set, first one involves using statistical models that assume some distribution and second one measures distances. Both methods assume that data points that are a considerable distance from any other cluster are outliers.

One variation of classification techniques is clustering. Even though, most classification techniques are based on labelled classes and these methods provides the ability to assign a new object to the most similar class, in real world applications sometimes the classes are not identified for large datasets. Clustering is the process of grouping data into classes (cluster), fulfilling two main rules. Any pair of objects within the same class has higher similarity and any pair of objects belonging to different clusters has lower similarity. Therefore, a cluster is a set of objects that are similar to other objects in the same cluster. Clustering can be used for outlier detection since it provides insight about data distribution and about the characterization of each cluster. Clustering belongs to several research areas like for example data mining, machine learning, marketing or spatial databases among others. Even though, clustering has been studied for many research fields, in this sub-section we provide an overview of clustering since the data mining side. Usually clustering in data mining has been challenged by the following issues:

- Scalability**:** Most clustering algorithms provides relatively good performance when working with small data sets (fewer than several hundred objects). On the other hand, in real world large databases contain millions or even billions of objects and current cluster methods might head to non-reliable results.

- Attributes source**:** Many clustering methods are intended to perform with numerical data, however sometimes algorithms are required to handle binary data, nominal data or ordinal data.
- Cluster shapes**:** Measures of similarity are usually based on Euclidean distance and Manhattan distance. These measures tend to identify cluster with spherical shapes and similar size. However, real world data distributions might not be grouped in spherical clusters, thus clustering should provide functionalities to manage clusters of different shapes.
- User's inputs**:** Some clustering algorithms are dependent of user's inputs. A user input might determine the desired number of clusters. Clustering methods are usually sensitive to user's inputs that might lead to biased results.

- Noise**:** Real world datasets usually contains noise and outliers that are difficult to handle for clustering methods that may produce poor quality clusters.

- Data dimensions: Even though, clustering techniques have proved good quality when data has two or three dimensions, most algorithms provides lower clustering quality when working with high-dimensional data. Furthermore, high-dimensional data might be sparse and even skewed.

2.3.1.1 Clustering methods

Even though, several clustering methods can be found in the literature and some of them might overlap with another method, this section provides an overview of main clustering techniques attending to how they built the clusters. Main typologies of clustering methods are, partitioning, hierarchical, density-based, grid-based, model-based, high-dimensional and constraints methods.

Partitioning clustering methods works under the following assumptions, given a dataset of n objects, the algorithm splits data into k partitions being $k \leq n$. Each cluster contains at least one object and each object belongs to exactly one cluster. However the last assumption can be relaxed for fuzzy distributions where data can be overlapped. Partitioning methods creates an initial partition depending on the desired number of clusters (k). The method tries to improve the initial partition using an iterative relocation technique that moves objects from one group to another depending on their similarities and dissimilarities trying to reach the optimal partitioning. To obtain the optimal partition, the clustering method would need to exhaustively explore all possible partitions. This approach provides the best solution but the worst performance. Therefore, most clustering methods employs a heuristic algorithm to obtain one of the best partitions. Most popular heuristics algorithms employed for such purpose are the k-means and the k-medoids algorithm. In the k-means approach each cluster is represented by the mean value of the objects in the cluster. On the other hand, the k-medoids algorithm represents each cluster using one of the objects in the cluster, probably the one that is closer to the centre of the cluster. These algorithms have showed good performance for finding spherical clusters in medium-sized data sets.

On the other hand, the k-means algorithm takes the input parameter k and splits a set of n objects into k clusters having higher intra-cluster similarity and lower inter-cluster similarity. Cluster similarity is measures using the mean value of the objects in the cluster, which is the centroid of the cluster. The algorithm firstly randomly selects k objects that form the clusters centres. For each remaining object, it is assigned to the most similar cluster based on the distance between the object and the cluster mean. Then the algorithm re-calculates each cluster centre. This process is repeated iteratively until some function converges, usually the square-error function. Square-error function is defined as follows:

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} |p - m_i|^2$$

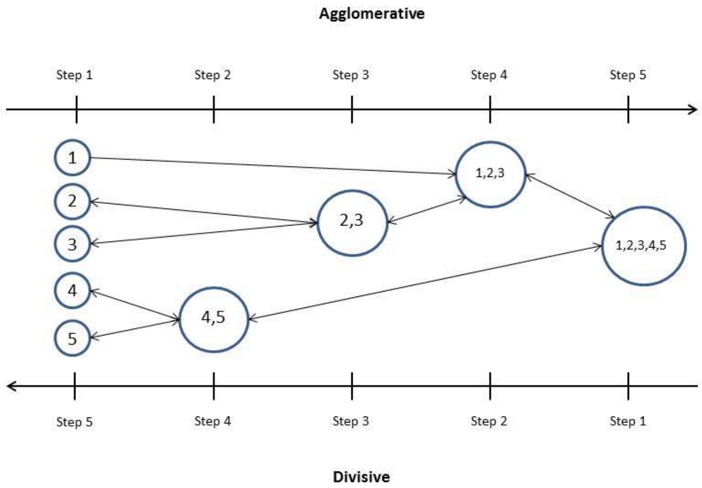
where E is the sum of the square error for all objects in the data set, p represents a given object and m_i is the mean of cluster c_i. K-means algorithm is sensitive to outliers; in fact one object with uncommon features might totally change the distribution of data. In addition, the square-error function far from alleviating these problems exacerbates the effect of outliers. To alleviate this problem, the k-medoids algorithm employs a representative point of some cluster to represent the cluster. Each remaining object is compared with the representative point of every cluster and it is assigned to the most similar. The algorithm tries to minimize the dissimilarities between each object and its representative point. For that purpose the absolute-error function is used:

$$E = \sum_{i=1}^{k} \sum_{p \in C_j} \left| p - \partial_j \right|$$

being E the sum of the absolute error for all objects in the data set, p represents each object in cj. The algorithm depends on the implementations but usually iterates until each point (p) is the medoid of its cluster. Though flat clustering methods have several advantages, they have many disadvantages that may be overcome using hierarchical clustering. Unlike flat clustering, hierarchical clustering provides a structure with more information and does not require pre-specifying the number of clusters. However these advantages of hierarchical clustering come at the cost of lower efficiency due to the higher complexity of hierarchical methods. Hierarchical methods rely on grouping objects into a tree of clusters. These methods can be classified as agglomerative and divisive depending on the approach to build the hierarchy, bottom-up (merge) or top-down (split). Main drawback of both approaches is the inability of hierarchical methods to backtrack and fix a poor split or merge decision.

Agglomerative clustering follows a bottom-up strategy where each particular object is placed in its cluster and then merges these clusters into a larger cluster until all objects fit in a cluster or some termination condition is reached. On the other hand, divisive clustering works under a top-down approach where each element is assigned to a large cluster, in successive iterations this clusters is split into smaller clusters. There is no clear consensus of which approach provides better clustering. However, flat clustering should be use when the number of clusters is known beforehand or when efficiency really matters. Moreover, hierarchical clustering is more suitable when the number of clusters is unknown or when the results needed requires more information than the cluster itself. Figure 26 shows main differences between those clustering types.

Figure 26 Agglomerative clustering vs divisive clustering



Finally density based methods proposes an alternative to most distance based methods previously explained. Even though, distance-based clusters perform well when finding spherical-shaped clusters, these methods suffer whit arbitrary-shaped clusters. For that purpose, other clustering methods have been developed based on the idea of density. The approach under the density-based methods is to continue expanding clusters as long as the number of elements in their neighbourhood exceeds some threshold. These methods are highly related with noise and outliers filtering. Main methods of density-based clustering are DBSCAN Ester, et al., (1996) and OPTICS Ankerst, et al., (1999).

- **Grid-based methods**

Grid-based methods are particularly useful when dealing with massive datasets. The principle is to summarize the dataset with cells that form a grid structure. Therefore, the approach consists of clustering the space surrounding the data points instead of clustering the data points themselves. The result is a grid summarizing the data points. This grid structure is then used to perform every cluster operation. The efficiency of these methods depends on the number of cells per dimension instead of the number of objects. Thus, efficiency is one of the most important advantages of grid-based clusters. Even though, first proposals of grid-based methods rely on rectangular grids, new approaches are arbitrary-shape capable. Main drawback of rectangular-shaped grids is that all data space is covered with the same precision, no matters the data density. Most commonly used grid-based method is called Statistical Information Grid Wang, et al., (1997). STING performs grid-based clustering dividing the spatial area into rectangular cells. It also provides information about the structure of the spatial space assigning each cell a level depending on the resolution and the density. Each cell at the higher levels is divided into new cells of lower levels. Once the grid structure is constructed, some statistical parameters (count, mean, standard deviation, minimum, maximum, etc) are pre-computed for the grids at the lower levels. These parameters can easily be calculated from the parameters of the lower level cells. This mechanism improves the efficiency of grid-clustering.

- **Model-based methods:**

Model-based clustering tries to fit every cluster with one model. The principle is search the best model that fits all data in the cluster. Model-based methods belong to flat clustering and usually employ density functions to identify the clusters. Most widely spread model-based algorithm is called Expectation-Maximization (EM) and is a generalization of k-means. Model-based algorithms perform clustering according to a model or probability distribution that fits the data points in the cluster. In fact the entire data space is a mixture of these distributions. Thus, main challenge of these algorithms is to determine the parameters that best fit the data. Even though, these algorithms used to converge rapidly, they could not reach the global optimal solution.

- **High-dimensional methods:**

Most data sets contain a large number of features or dimensions. In fact most text documents contain a huge amount of terms or keywords that require clustering methods able to deal with high dimensionality data. As the amounts of dimensions grow the data becomes sparse due to the meaningless of distance measures between pairs or due to the low density of every data space area. Clique and Proclus are able to deal with this type of data by exploring a subset of features rather than performing over the whole data space.

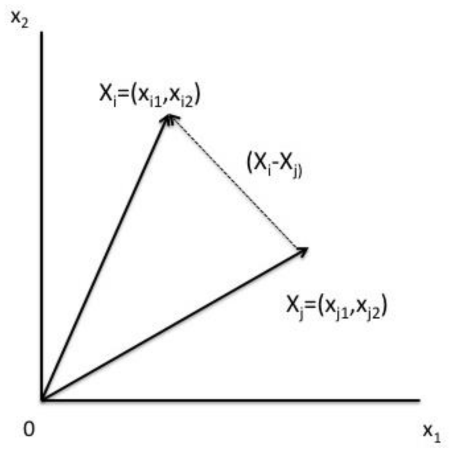
2.3.2 Finding patterns

Data mining system produce large set of patterns or rules that leads to the idea that not all patterns will be relevant or interesting. Furthermore, just a reduced subset of patterns will be interesting for applications. To identify whether some pattern is interesting or not, some objective rules exists. These rules are based on probabilities that compute the percentage of transactions in the dataset that the rule or pattern is satisfied. However, objective rules must be combined with subjective rules since the relevance of some patterns depends on the user preferences and interests. Even though, most data mining systems have the ability to obtain most interesting patterns, this is not realistic and applicable due to performance constraints. Usually the user provides some restrictions to the search and the completeness of the data mining algorithm is measured taking into account that constraints.

2.3.3 K-nearest neighbours

The k-nearest neighbour's algorithm is a non-parametric method where no parameters are estimated. The algorithm employs the proximity of the input observations in the training set and their corresponding outputs to predict the proximity of the objects in the test set. The output variables can be interval variables in which case the algorithms is used for prediction or can be nominal or ordinal variables in which case the method is used for classification. Before going into the details of the algorithm it is necessary to define the Euclidean distance. Euclidean distance between two vectors of items x_i and x_j is calculated as follows. Assuming the case of two dimensions the vectors can be represented as $x_i = (x_{i1}, x_{i2})$ and $x_j = (x_{j1}, x_{j2})$. Figure 27 depicts the scenario.

Figure 27 Euclidean distance



Formally the Euclidean distance is computed as follows:

Formula 1 Euclidean distance computation two dimensions

$$d(x_i, x_j) = |x_i - x_j| = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2}$$

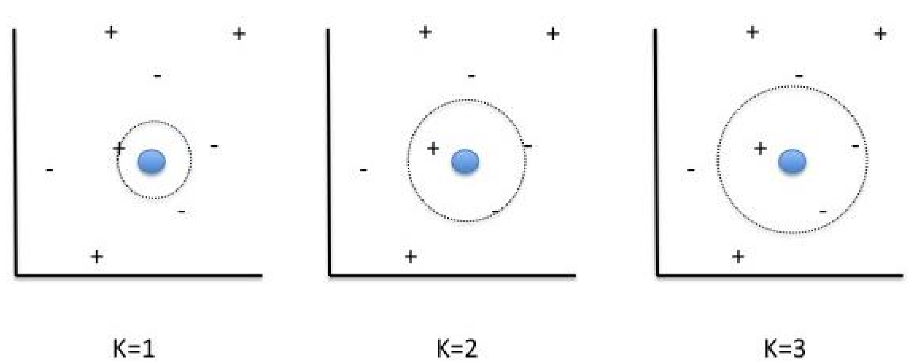
The previous definition in formula 1 can be extended to the general case of N dimensions. The formulation is the following:

Formula 2 Euclidean distance computation N dimensions

$$d(x, y) = |x - y| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}$$

Taking into account that the k-nearest neighbour is dependent on distance measure it is desirable that input data was standardized. To get an idea of how the algorithm performs, Figure 28 depicts the scenario of finding the k-nearest neighbours with k ranging from 1 to 3.

Figure 28 K-nearest neighbour's representation



As can be seen in the previous figure, for k=1, the result is plus, for k=2 the result is unknown and finally for k=3 the result is minus. Furthermore, Figure 28 depicts the scenario of predicting the output value for a vector in the test set $x_a = (x_{a1}, x_{a2})$, which associated output value is $y_a$and it's the estimated value. In addition, Figure 28 shows three representative points in the training set, $x_i = (x_{i1}, x_{i2})$, $x_j = (x_{j1}, x_{j2})$, $x_k = (x_{k1}, x_{k2})$ and their associated outputs y_i, y_j, y_k. In Figure 29 it can be seen that x_i is the closest data point (closest neighbour), that is $d(x_a, x_i) < d(x_a, x_j) < d(x_a, x_k)$. Therefore, having k=1, the estimated value $y_a = y_i$, which is the output value of its nearest neighbour.

Figure 29 Predicting test set output using nearest neighbours

Formally the problem of finding the k-nearest neighbours having $P = \{\overrightarrow{p_i}\}_1^n$ a set of n points in a d-dimensional space, and let $\vec{q}$ be the estimated point, can be stated as follows:

Find the point $\overrightarrow{p_c} \in P$, which is the minimum distance from $\vec{q}$.

Figure 30 K-nearest neighbour's formal definition

$$\|\vec{q} - \overrightarrow{p_c}\| \leq \|\vec{q} - \overrightarrow{p_i}\| \; \forall \, \overrightarrow{p_i} \in P$$

Figure 30 details the mathematical formalization of K-nearest neighbours. This is a general formulation that is usually modified according to the needs of the problem.

3 State of the Art

This section provides a short view of user models researches since the beginning of this paradigm till the latest approaches. Recommender systems became important with the appearance of the first papers on collaborative filtering in the middle of the nineteen's Resnick, (1994). The area of recommender systems has been of great interest during the last decade due to abundance of applications that help users with personalized content. Thus, much work has been done in this area from the industry and the academia. Examples of these applications that help users providing personalized content and recommendations are, Amazon.com recommending books or CDs Linden (2003), or Versifi recommending news Billsus, et al., (2002). In addition some commercial applications have incorporated recommendation capabilities into their commerce servers to provide their customers personalized suggestions Peddy and Armentrout, (2003). Even though first recommender systems date from the mid-nineteen's and several advances have been developed, current generation of recommenders still needs improvements to make recommender systems applicable for more real-life applications. Some studies reveal that recommender systems are still far of being applicable and effective in some fields Wade, (2003).

First approaches to recommender systems belong to researchers in Natural Language and Dialogue Management classifying users Perrault, et al., (1978). It was during the mid-eighties when distinctions between components of the recommender system and components of the application were done. User modelling via stereotypes was already proposed in late seventies Rich., (1979), and during the mid-eighties the group user models technology arises, with systems like GUMS Finin & Drager, (1986). It was in the mid-nineteen's when recommender systems emerged as an independent research area and when researchers started to focus on recommendation problems and explicitly rely on ratings.

The problem of recommenders on its simplest form can be seen as estimating a rating for some item that has not previously been seen for the user. This definition of recommender problem was formalized as follows; Let C be the set of all users and let S be the set of all items that can be recommended, both spaces (C and S) might be very large. Let u be a utility function that measures the usefulness of item s to user c.

Let the utility function be $u: C \times S \rightarrow R$

Where R is totally ordered Adomavicius and Tuzhilin, (2007). Most recommender systems represent the result of the utility function as a rating that specifies how much a user likes some item. Each element of the user space can be described with a profile including main features of the user. The simplest profile just consists of a user identifier. Likewise each element of the item space is defined with a set of features. Therefore, main problem of recommenders is related to utility function that is usually not defined on the whole C x S space. In fact most recommenders need to extrapolate their utility function to reach the whole space C x S. Since the utility function is usually characterized by ratings, it is initially defined over the set of items rated. Therefore, the recommender should estimate that ratings for each non-estimated pair user/item and provide reliable recommendations based on these estimations. Estimations are usually done by heuristic methods or by optimizing some criterion like for example the mean square error when predicting a rating. Once these unknown ratings are estimated for current user the system must select the highest one or the N highest ones depending on the problem.

In addition two different typologies of recommenders can be found, rating-based recommenders and preference-based recommenders. First approach predicts the absolute value of ratings while second one focus on predicting a proper order of items for some current user Cohen, et al., (1999) Jin, et al., (2003).

Even though main classification of recommender system is done concerning the approach used to estimate ratings, another classification can be done according the purpose of the user model. Thus main user models have been developed in the academic area or in the industry area Kobsa, (2001). In addition, main focus of this thesis is statistical user modelling, which includes most user model or recommender systems.

3.1 Academic developments and Industry developments

First approaches to user models enable creating hierarchies of stereotypes and relate a set of rules to each stereotype. An example of these first approaches is General user modelling shell (GUMS) Finin, (1989). The sets of rules that GUMS handles describe the behaviour of each user in its knowledge base. In addition, GUMS provides features to store new facts and verify the consistency of new rules against the old ones. To maintain the consistency of the rules, GUMS store facts in a tree structure and updates such structure every certain time. In case that soma inconsistency cannot be solved, the stereotype responsible for that inconsistency will be removed from the database.

At the end of the eighties, some regulations were established to user models, these regulations define a set of rules that a user model must meet Kobsa, (2001):

- Representation of assumptions about one or more user characteristics
- Representation of users in subgroups including features of these users in the user model
- Recording of user's past interaction with the system
- Formalization of assumptions based on interaction history
- Consistency maintenance in the user model

Therefore, at the beginning of the nineteen's the User Modelling Tool (UMT) Branjnik & Tasso, (1994) appears. UMT is also classified within the stereotype user models. UMT provides a maintenance system and user's information can be classified as premises or assumptions. Some other tools like Prolog based Tool for User Modelling (PROTUM) Vergara, (1994), or Theory and Applications for General User/Learner-modelling Systems created by Paiva and Self (1994) emerge like new approaches of academic user models. In addition BGP-MS (2001) (Belief, Goal and Plan Maintenance System) created by Kobsa and Pohl allows creating assumptions about the user and recording past user session. Finally two more approaches come up meeting most requirements of academic user models. These tools are TAGUS (1994) created by Paiva and Self and Doppelganger that was developed by Jon Orwant, (1995). Both tools provide system maintenance, representation of user's assumptions and a knowledge base full of knowledge of past sessions.

At the end of the nineties, the concept of web personalization was increasingly recognized in the area of electronic commerce. In fact web segmentation provides product offering and sales promotions to be targeted to each particular user. Thus, taking user navigation habits and previous interactions, it is possible to migrate from mass marketing to one-to-one marketing.

The first approach of web oriented user models is Group Lens, developed by Net Perceptions Breese, et al., (1998), Herlocker, et al., (1999). As mentioned before Group Lens is a web oriented user model, based on ratings of user's behaviour when surfing the web. The second approach arises when Andromeda develops LikeMinds (2000). LikeMinds is pretty similar to GroupLens, the main difference is in its architecture and that GroupLens provides ODBC support. Another approach to web oriented user models is the Personalization server that appears in 2000. The tool is based in pre-fixed stereotypes. In addition, the tool classifies new users within these stereotypes according to their demographic information. Lately the Personalization server has been used to personalize web pages and it has been classified within classical user models. Finally two more approaches of web oriented user models appear during the first decade of the 21st century. These tools are called FrontMind and Learn Sesame. Both tools are based on pre-fixed stereotypes and have been applied for several personalization purposes.

3.2 State of the Art in Statistical User Models

Statistical user models are concerned with the use of observed sample results and provide predictions about unknown dependent parameters. In the case of user modelling these parameters are aspects, likes, preferences or behaviours of a user and represent his or her goals. According to Larson "Statistical user models are intended to perform predictions of a dependent parameter, departing from observed samples. Such parameter represents an aspect of a user future behaviour" Larson, (1969).

A well-known problem within Artificial Intelligence community is about obtaining the user modelling systems knowledge bases and it is called the bottleneck problem. This problem affects group user models that rely on hand-built knowledge bases. In addition, hand-building a knowledge base with many representative points is harder task than hand-building a knowledge base with just a few points to consider. Furthermore, some other factors might affect group user models, for instance vitiated data together with certainty degradation.

Therefore, main problem of group user models can be related to the bottleneck problem, this problem together with the amount of data generated by recent application domains pointed towards statistical user models as an encouraging alternative to the traditional approach. In addition, large quantities of electronically available data together with the advances performed in machine learning boost the use of statistical models instead of group user models.

Recently, due to the information explosion and the improvements in communications people have access to large repositories of information almost anyplace and anytime. In addition, information providers can figure out more information about user by monitoring people's activities. The mixture between electronic content and knowledge about users leads to statistical user models as an encouraging trend to build user models that support the delivery of personalized content.

The usage of statistical and probabilistic user models has been manifested in UMUAI papers during the last years, showing a clearly growth as years go by. In particular, noteworthy articles in UMUAI have identified several challenges that user modelling presents to statistical and probabilistic modelling techniques Albrecht and Zukerman, (2007). These challenges can be classified in three different categories: limitations of current user modelling approaches, dynamic nature of user modelling data and efficiency considerations.

Recommender systems have been classified according to their approach to estimate ratings, most accepted classification were done by Balabanovic and Shoham, (1997):

- Content-based recommenders
- Collaborative recommenders
- Hybrid systems

3.2.1 Content-based approach

In the content-based approach, the behaviour of a user is predicted from his or her past behaviour, content-based approach has its roots in Information retrieval Baeza-Yates, Ribeiro-Neto, (1999). And so then, first recommenders under the content-based approach involve retrieving textual information. Main improvement over Information Retrieval approaches comes from the use of profiles that store user preferences and likes. Content-based recommenders employ survey techniques or implicit methods to obtain these profiles.

Thus, the utility function u(c, s) of items s for user c is estimated based on the utilities u(c, s_i). Where s_i is the set of ratings assigned by user c to items similar to s, then $s_i \in S$. Content-based learning is used when users past behaviour are reliable indicators of their future behaviours or preferences. Therefore, user models based on this approach are built departing from the data of user's past behaviours. Content-based approach is suitable for scenarios where users behave in some manner distinctively. On the other hand, this approach requires that systems collect large amounts of data from each user to perform statistical predictions. Another disadvantage of content-based approach is that the features selected when building a content-based model have a substantial effect on the usefulness of this model Zukerman and Albrecht, (2001). In fact very specific features make systems useful only for repetitive behaviours. On the other hand, too general features provide poor quality predictions. As previously mentioned most recommenders under the content-based approach, recommend items basing on textual information such as keywords. Therefore the problem focuses on determining the importance of word k_i in document d_j. Such importance is determined using a weighting measure; most common measure is the Term Frequency Inverse Document Frequency (TF-IDF) Salton, (1989). TF-IDF is defined as follows, assuming the whole set of documents N and assuming that the keyword k_i appears in n_i document of N. Let f_{ij} be the number of times that keyword k_i appears in document d_j, then the TF_{ij} of keyword k_i in document d_j is measured as follows:

$$TF_{ij} = \frac{f_{ij}}{max_z f_{zj}}$$

Where the maximum is calculated over the frequencies f_{zj} of all keywords k_z that appear in document d_j. Even though, this is one of the most used metrics it lacks when keywords appear in most documents, then another method must be combined with the TF-IDF. Thus recommender systems under content-based approach will suggest items previously rated by current user Lang, (1995). On the other hand, Bayesian networks were also used to determine whether an item is relevant or not for some concrete user Pazzani and Billsus, (1997). In addition, the Winnow algorithm was used for the same purpose Littlestone and Warmuth, (1994). However Winnow algorithm was preferred to TF-IDF or Bayesian networks when there are many features per item or per user Pazzani, (1999).

There are some limitations that recommenders under the content-based approach must face to demonstrate accuracy in its predictions.

3.2.1.1 Partial content analysis

Content-based recommenders produce suggestions according to the features linked to the objects they recommend. Thus, a computer must parse content or features must be extracted and associated to an object manually. In certain domains it might be difficult to automatically extract those features. While it is easy to obtain features from textual resources, it may be difficult to manage from multimedia resources. In addition, in textual resources if two documents have the same set of keywords these two documents become undistinguishable for the recommender Shardanand and Maes, (1995).

3.2.1.2 Over-specialization

Content-based recommendations are restricted to items previously rated. In order to ease this problem the most accepted solution involves introducing some randomness. For example the use of genetic algorithms was proposed as a solution Sheth and Maes, (1993). However over-specialization problem not only involves recommending items very different to those previously rated by the user, but also involves the problem of recommending items too similar to those rated by current user. Therefore, some authors introduce the idea of redundancy in content-based systems Zhang, et al., (2002).

3.2.1.3 New user problem

New user problem in recommenders affects those users with insufficient number of ratings. System needs sufficient information about the user to produce reliable recommendations.

3.2.2 Collaborative approach

In the collaborative approach, a user's behaviour is predicted from the behaviour of other like-minded people. Thus, collaborative learning is used under the assumption that a user behaves similarly to other user. Collaborative systems are built using data from a group of users and achieving predictions over an individual user. Main limitation of systems under the collaborative approach is that they make predictions about the behaviour of a single user departing from observations of many users, hence they do not make predictions tailored to individual users. On the other hand, collaborative learning systems are more suitable than content-based systems when predicting about a new user or predicting about a known user under a new situation. As a matter of fact, collaborative filtering has been used over the years to build successful recommender systems Xiaoyuan and Taghi, (2009). Usually collaborative filtering systems have a user-item or user-feature matrix. Table 2 shows an example of user-feature matrix for collaborative learning which most time is implicit in collaborative system knowledge bases.

Table 2 User-feature matrix

	Shrek	Snow White	Spider-man	Super-man
Alice	like	Like		Dislike
Bob		Like	Dislike	Like
Chris		dislike	Like	
Tony	Like		Dislike	?

There are some missing values in table one where users did not give their preferences.

Analogous to content-based approach, models under collaborative approach must deal some limitations to provide reliable predictions.

3.2.2.1 Data sparsity

Data sparsity domains can challenge the quality of the recommendations provided by some recommender. One specific case of the data sparsity problem is the cold start problem also known as the new user problem or the new item problem Yu, et al., (2004). New items cannot be recommender because they have not been rated and new users cannot be classified due to the lack of information about these users. Some other problems are related to the sparsity problem in recommender systems, for instance the reduced coverage problem. Coverage has been defined as the percentage of items that can be recommended Xiaoyuan and Taghi, (2009). Then, the reduced coverage problem arises when the percentage of pair's user-rating is too low compared with the number of items in the system. In that case, the recommender won't be able to provide predictions over the whole set of items in its knowledge base. Last problem associated to data sparsity is called the neighbour transitivity problem. This problem happens in sparse databases when two similar users are identified as very dissimilar users just because they both have not rated the same items Bessa, et al., (2012).

To alleviate the data sparsity problem, dimensionality reduction techniques Billsus and Pazzani, (1998) have been used. Several methods belonging dimensionality reduction were used, most important are. Singular Value Decomposition, which removes unrepresentative data-points from the user-feature matrix. Latent Semantic Indexing (LSI) Landauer and Littman, (1994) that calculates similarity among users in a reduced space instead of considering the whole users-items space. Finally, Goldberg, et al., (2001) developed eigenstate that also employs dimensionality reduction techniques to alleviate the sparse data problem. On the other hand, some authors identified that applying dimensionality reduction some information about users and items will be removed from the system leading to some degradation in the quality of the recommendations Sarwar, et al., (2000).

3.2.2.2 Scalability

Scalability problem happens when the number of users and items starts growing. Collaborative filtering algorithms suffer the problem of scalability reducing their performance as the amount of information grows. In fact, some collaborative algorithms are intended to provide real-time responses to their uses, for instance those systems that need to react to online requirements. Therefore, those systems request high scalability and perform the most time-consumption calculations offline Linden, et al., (2003). In addition, Linden, et al., provides a comparison of how traditional methods handle scalability.

Traditional collaborative learning algorithms do not provide any offline computation and its online calculations scales with the number of users and items. By the way, this approach does not provide any solution to scalability problem unless it uses dimensionality reduction or partitioning, but in these cases this will affect to the quality of the recommendations. Even though clustering performs much of the computation offline, it has been proved that these recommenders provide poor-quality predictions. Finally, the authors identify that search-based methods provide poor scalability features. Thus, it can be stated that the clue to provide scalable systems is in the offline computation and concretely in the ability of the system to build and reorder its knowledge base offline.

3.2.2.3 Synonymy

Synonymy occurs when several concepts or items have different names or entries. Most recommender systems under the collaborative approach are unable to discover these relationships yielding debatable predictions when synonymy occurs. For example the concepts "children movie" and "children film" refers to the same but most system would find no match between them. Therefore, the prevalence of synonyms decreases the recommendation performance of collaborative filtering systems. Most attempts to solve the synonymy problem depend on term expansion or the construction of a thesaurus, however fully automatic methods are still non-applicable due to its rapid degradation Jones, (1972). Once again, some dimensionality reduction methods have been used to ease the problem of synonymy. The Latent Semantic Indexing (LSI) method (See 3.3.1.2.1.1) is able to alleviate the synonymy problem ignoring the less important terms. However, it has been proved that the performance of LSI method is quite encouraging at higher recall levels. On the other hand, its performance seems to decrease at the lowest levels of recall Deerwester, et al., (1990). Another problem identified by Deerwester, et al., and related to synonymy is the polysemy, which means that most words have more than one meaning. The author's state that LSI method deals properly with the synonymy problem but it provides poor capabilities to deal with polysemy.

3.2.2.4 Grey Sheep

The name of Gary Sheep problem or Gary Sheep users refers to users whom do not totally agree or disagree with any group and are difficult to classify for collaborative filtering systems

Claypool, et al., (1998) Burke, (2002). In recommender systems there might be users that have rare likes or preferences compared to the rest of the community and that are difficult to classify for collaborative filtering methods Ghazanfar and Prugel-Bennett, (2011). In fact these users usually have a low correlation degree with other users. In addition, Mustansar and Prügel-Bennett, (2014) identified that the presence of these users in a medium community of users leads to two different problems: first one is that these rare users will not receive accurate predictions, second one is that these users may affect negatively to the rest of the recommendations. Even though, the problem of grey sheep has not been widely addressed in the literature, an algorithm to identify was proposed. Mustansar and Prügel-Bennett proposed this algorithm and it is inspired in shape detection during image processing. According to these authors same procedures applied to separate rare shapes in a different cluster can be employed to detect grey sheep users and separate them into a different cluster. The proposed algorithm departs from the K-means++ clustering algorithm Arthur and Vassilvitskii, (2007). This algorithm was modified to cluster user-item pairs and detect grey sheep users. The basics of the proposed algorithms are in calculating the top-rated users that is users that have rated larger number amounts of items. Finally use these users to calculate the centroids of the clusters and identify the grey sheep users. Furthermore, some users identify the black sheep problem, which refers to users that due to his idiosyncratic are almost impossible to classify, however this problem has been considered for most authors as an acceptable failure McCrae, et al., (2004).

Regarding the ways employed to build recommender systems, two main approaches are well-known in the literature, the Memory-Based and the Model Based approach.

3.2.2.5 Memory-Based Collaborative Filtering Techniques

Memory-based collaborative filtering algorithms use the whole or a set of the user-feature database (user-feature matrix) to compute similarities among users or items. Every user is a part of a group that consists of several users with same likes or preferences. Therefore, this type of algorithms requires calculating the neighbours of each user. This calculation is based on the neighbourhood-based algorithm, which uses the following steps: it calculates the similarity or correlation between two users, it produces a prediction for current user based on the weighted average of all the ratings of the user or simply using a weighted average Sarwar, et al., (2001). On the other hand, when the system needs to produce the top-N recommendations, the algorithm looks for the k most similar users (nearest neighbours) after calculating the similarities. Therefore, the similarity calculation is a crucial step in memory-based algorithms. Similarity can be calculated for two different types of systems, item-based systems and user-based systems. The main idea to compute the similarity between item i and item j is in retrieving the users who have rated both items and then employ a similarity measure to determine the similarity w_{ij}. There exist several measures to compute this similarity between items or users, the most important are Pearson correlation and Vector Cosine similarity. Pearson correlation measures how much two variables relate each other. The general expression for Pearson correlation is depicted in Figure 31.

Figure 31 Pearson Correlation

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

This general expression can be applied to calculate how two users (u, v) relate each other. For that purpose the first summation, must go over the whole set of items that both users u and v have rated and $\bar{X}$ is the average rating of the co-rated items for user u, while $\bar{Y}$ is the averaged rating of the co-rated items for user v.

On the other hand, with the vector cosine method the similarity is measured considering each document as a vector of word frequencies and computing the cosine of the angle formed by the vectors Salton and McGill, (1983). This paradigm can be borrowed to collaborative filtering changing documents for users or items and frequencies for ratings. Figure 32 depicts how vector cosine-based similarity is computed.

Figure 32 Vector cosine-based similarity

$$similarity = \cos(\theta) = \frac{A * B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} * \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

Having the (m x n) user-item matrix, the similarity between two items i, j is calculated as the cosine of the vectors corresponding to the columns i and j of the matrix.

3.2.2.5.1 Computing recommendations

Produce recommendations is the most important task in collaborative filtering. Taking into account the neighbourhood-based algorithm, the predictions are calculated using the ratings of the most similar user. To aggregate those ratings, two main approaches have been used, weight the sum of neighbour's ratings and use a simple weighted average. To produce a prediction for current user a on some item i, all ratings on that item can be weighted according to the following formula Resnick, et al., (1994).

$$P_{a,i} = \bar{r_a} + \frac{\sum_{u}^{U}(r_{u,i} - \bar{r_u}) * w_{a,u}}{\sum_{u}^{U}|w_{a,u}|}$$

where the summations are done over all users $u \in U$ who have rated item i. r_a and r_u are the average ratings for the user a and the user u. Finally $w_{a,u}$ are calculated using the vector cosine-based similarity. On the other hand, in order to compute predictions, the simple weighted average can be employed Sarwar, et al., (2001). The simple weighted average for an item-based prediction can be calculated as follows:

$$P_{u,i} = \frac{\sum_n^N r_{u,n} w_{i,n}}{\sum_n^N |w_{i,n}|}$$

where in this case summations are over all rated items for user u, $w_{i,n}$ is the weight between items i and n. Finally $r_{u,n}$ is the rating for user u on item n.

3.2.2.5.2 Computing Top-N recommendations

The top-N recommendation involves producing an ordered set of items according to current user preferences. Top-N recommendation algorithms analyse the user-item matrix to figure out relations between users and items and provide suggestions. Two approaches have been developed to produce the top-N recommendations in collaborative filtering systems. First approach is the user-based Sarwar, et al., (2000), this method relies on the idea that each person can be classified in a larger set of similar users having similar behaviours. Finally, items liked by most people in that group can be used to produce reliable recommendations. Second approach is known as item-based Kitts, et al., (2000) and it basics are, analyse historical information to identify relations between items. Such relations might be for example that the purchase of some item always leads to another item purchase. The user-based approach suffers serious problems with scalability and real-time performance Karypis, (2001). On the other hand, the item-based method alleviates the scalability problem but must deal with the problem of finding sub-optimal solutions Deshpande and Karypis, (2004).

3.2.2.6 Model-Based Collaborative Filtering Techniques

The advance in the areas of machine learning and data mining enables the development of models able to recognize patterns in a training set and make intelligent predictions for real world (test set). Model-based collaborative filtering algorithms have been investigated to alleviate main problems of memory-based algorithms Breese, et al., (1998). Several techniques have been used over the years to build model-based systems. Main techniques are Bayesian networks, clustering algorithms and regression-based algorithms. The simple Bayesian algorithm applies a Naïve Bayes to perform collaborative predictions. The Naïve Bayes assumes that all features are independent according to its class, then the probability of a class given the features can be computed and the class with highest probability is classified as the predicted class Miyahara and Pazzani, (2002). In addition, clustering techniques can be used to build user models under the model-based approach. A cluster can be defined as collection of objects that are similar to other objects in the same cluster but are dissimilar to the objects in other clusters Han and Kamber, (2001). Clustering algorithms usually employs the Minkowski distance Groenen, et al,. (2007) or the Pearson Correlation Herlocker, et al., (1999) to compute the similarity among objects. The Minkowski distance for two data objects $X = (x_1, x_2 \dots x_n)$ and $Y = (y_1, y_2 \dots y_n)$ is normally calculated according to the following formula:

$$d(x,y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}$$

Where, n is the dimension of the object and x_i and y_i are the corresponding values of the element in the position I of the initial data objects. Finally q is a positive integer that may take two values, if $q = 1$ then $d(x, y)$ is Manhattan distance, if $q = 2$ then $d(x, y)$ is the Euclidean distance.

Clustering methods can be classified in three different categories: partitioning methods, density-based methods and hierarchical methods Han and Kamber, (2001). The most extended partitioning method is k-means created by MacQueen, (1967). This method has two main advantages, it is relatively efficient and soft implementation MacQueen, (1967). Density-based methods usually search for dense areas or cluster and separate them from sparse areas that represent noise. Main examples of clustering methods are DBSCAN Ester, et al., (1996) and OPTICS Ankerst, et al., (1999). On the other hand, hierarchical methods splits the whole set of objects or user building a hierarchical decomposition according to some criterion. Main example of hierarchical methods is BIRCH Zhang, et al., (1996). Usually clustering methods are used to produce a set of clusters that are used for later analysis. For example clustering can be applied to split the data into clusters and after that use some algorithm to provide predictions within some of the clusters O'Connor and Herlocker, (1999).

3.2.2.6.1 Regression algorithms for model-based collaborative filtering

Some algorithms employed in memory-based systems might determine that two vectors are really distant based on Euclidean distance, however these vector may be really similar in terms of other metrics like Pearson correlation. In those scenarios, memory-based algorithms are not able to accurately determine whether the vectors are similar or not. Regression methods can help to ease this problem. Having $X = (x_1, x_2 \dots x_n)$ being a random variable that represent user's preferences, a regression can be formulated as follows:

$$Y = \begin{bmatrix} Y_{00} & \cdots & Y_{0n} \\ \vdots & Y & \vdots \\ & \cdots & Y_{mn} \end{bmatrix} = \begin{bmatrix} A_{00} & \cdots & A_{0n} \\ \vdots & A & \vdots \\ & \cdots & A_{kn} \end{bmatrix} \begin{bmatrix} B_{00} & \cdots & B_{0k} \\ \vdots & B & \vdots \\ & \cdots & B_{mk} \end{bmatrix} + N$$

Where N = ($N1$,….., N_n) is a random variable representing noise. Each Y_{ij} is the rating of user i on item j. Each X_{ij} is an estimation of the variable X. Main problem of this approach is that usually the matrix Y suffers the sparsity problem. A solution to this problem was provided by Canny, (2002) who introduced the idea of *sparse factor analysis,* which replaces missing elements (those elements with the average of non-missing elements) of the matrix Y.

3.2.3 Content-based learning versus Collaborative learning

Content-based learning is used under the assumption that each user exhibits a behaviour under a given set of circumstances and that this behaviour should be repeated under similar circumstances. Moreover collaborative learning is based on the idea that a particular group of users tend to behave in similar way under some given circumstances. Over the years several comparisons have been performed between both content-based and collaborative approaches. One of the first comparisons was done over the domain of films recommendation. The comparison consists of checking the performance of two recommenders, first one built under content-based approach and exactly same recommender built under the collaborative approach Alspector, et al., (1997). Results yielded shows that models under the collaborative approach perform significantly better than those under the content-based approach. Even thought, the results yielded by system under the collaborative approach were better than those provided by the system under the content-based approach, the collaborative system present signs of over fitting. In addition, Alspector, et al. identified some restrictions of both approaches. Concerning collaborative approach, it can be used with user's that have not been assigned to a group. On the other hand, content-based approach requires careful feature selection (specialist intervention). Finally, study results indicate that this type of systems must combine both approaches (hybrid system) Burke, (2002).

3.2.4 Hybrid collaborative filtering techniques

Hybrid recommender systems combine collaborative learning techniques with other recommendation method to gain better performance. Usually collaborative filtering is combined with content-based filtering. In addition, some other systems combine collaborative filtering with demographic recommenders based on profiles Krulwich, (1997).

Content-based recommender systems analyse the context of information to provide recommendations. Then the content-based system uses heuristics or classification methods to make recommendations Pazzani and Billsus, (1997). However, content-based techniques suffer the sparse data problem (see 2.3.2.1) and require enough information to build a reliable classifier. In addition, they are restricted by features, which are associated explicitly with the objects. These features might be difficult to obtain, while collaborative filtering can make predictions without any descriptive data. Furthermore, content-based techniques suffer overspecialization, which restricts recommendations to items similar to those rated by current users Adomavicius and Tuzhilin, (2007). Attempting to avoid main limitations of content-based systems and improve recommendations, collaborative filtering and content-based filtering are combined to build hybrid recommenders. Several classifications of hybrid recommender systems can be found along the literature, however the most extended one is that one done by Burke, (2002). According to Burke, hybrid recommender can be classified depending on the hybridization method, main approaches are, weighted, switching, mixed, feature combination, cascade, feature augmentation and meta-level. Hereafter, an overview of these approaches will be provided.

3.2.4.1 Weighted hybrid recommenders systems

Weighted recommender systems compute recommendations according to the results of all available recommendation methods in the system. In this case, the simplest systems would be a linear combination of recommendation scores, for example the P-Tango system Claypool, et al., (1999). P-Tango firstly assigns same weight to collaborative than to content-based algorithms but it continuously adjust weights according to the success of each prediction method. Main advantage of weighted hybrid recommender systems is that all system capabilities take part on predictions and it is easy to modify the weights to achieve more reliable recommendations. On the other hand, main weakness of these systems is that it works under the assumption that weights are uniform for each method independently of the item in the space of items.

3.2.4.2 Switching hybrid recommenders systems

Switching hybrid recommenders employs just one technique at the same time to produce recommendations. Most switching systems applies one method to provide a recommendation and in case that this method could not provide a sufficiently reliable recommendation switches to another method. For example the DailyLearner Billsus and Pazzani, (2000) uses a content-based/collaborative switching system. It first tries the content-based approach and if this approach cannot provide reliable recommendations switches to the collaborative filtering. This switching does not alleviate the sparse data problem (see 2.3.2.1) since both approaches suffers this problem. However, DailyLearner content-based method is the nearest neighbour Fix and Hodges, (1951), which does not require a large amount of samples to provide reliable classifications. In the case of DailyLearner, the switching to collaborative filtering provides the system the ability to recommend items that are not close in a semantic way to previously rated items. That is provide predictions about items that the content-based approach is unable to achieve. However the methodology of DailyLearner imposes that the content-based approach will be used first always. However, some alternatives to this plan have been proposed, for example use the agreement between user's ratings or likes and the recommendations of each technique to select the most suitable technique for next recommendations Tran and Cohen, (2000). Switching hybrid systems introduce a new challenge for user modelling community since the switching criteria must be defined and this creates a new level of parameterization. However, combining two recommendation techniques provides the system with main advantages of both methods.

3.2.4.3 Mixed hybrid recommenders systems

In some cases systems are required to provide large amounts of recommendations at the same time. In these cases it would be interesting to use a "mixed" approach where recommendations provided by every technique are presented together. Over the years some systems using this mixed approach have been developed, for example PTV system that uses a mixed approach to suggest a television program Smyth and Cotter, (2000). Content-based approach focuses on the textual descriptions while collaborative filtering is used to compute the preferences of other similar users. Final recommendation is based on the predictions of both approaches. Mixed approaches alleviate the sparse data problem (cold data problem) since the content-based filtering can be relied to propose recommendations of items not rated. However, the mixed strategies does not get around the new user problem since both, content-based and collaborative filtering need some background about current user to make reliable predictions.

3.2.4.4 Feature combination hybrid recommenders systems

Feature combination systems use collaborative information to assemble an augmented data set to be used with content-based learning. For example experiments using the prototype Ripper to recommend movies accomplished a significant improvement in precision over a simply collaborative approach Basu, et al., (1998). However, hand-filtering content features enables such improvement. In addition, the author identify that without hand-filtering content features, the system improves it recall but not the precision. Main advantage of feature combination is that it enables the system to consider collaborative information but it does not only rely on this information.

3.2.4.5 Cascade hybrid recommenders systems

Cascade hybrid systems relies on a staged process where one recommendation technique is applied first to obtain a set of candidates and a second method refines the recommendation from among the initial set of candidates. A well-known recommender system based on cascade methods is EntreeC Burke, (2002). The system employs its knowledge of restaurants to make recommendations. Recommendations done by the content-based approach are placed in buckets and collaborative filtering is used to break ties. Cascade systems main advantage is its efficiency since these systems are not forced to use the second, lower-priority mechanism. However these systems rely on its higher-priority mechanism since lower-priority techniques can only refine recommendations.

3.2.4.6 Feature augmentation hybrid recommenders systems

Feature augmentation methods apply one technique to produce a rating of items to be recommended, and then this information is fed to second recommendation method. Unlike to cascade methods, in this case the second recommendation technique is used in every recommendation process. For example Libra Mooney and Roy, (1999) makes content-based recommendations of books departing from information obtained by Amazon.com collaborative filtering. Feature augmentation provides additional functionalities added by intermediates and that can be used by other techniques. Note the differences among augmentation and the techniques presented before. Augmentation is different from feature combination since the last technique combines information from different recommendation methods. On the other hand, both cascade and augmentation techniques sequence two recommenders but in augmentation the input of second recommender includes the output of the first one, in cascade second recommender, which is a lower-priority recommender, does not use any output of first recommender.

3.2.4.7 Meta-level hybrid recommenders systems

Unlike feature augmentation that feeds part of the model generated by first recommender to second recommender, Meta-level systems feed the whole model to second recommender. First system supporting this approach was Fab Balabanovic, (1998). Fab uses content-based filtering using Rocchio's method Rocchio, (1971) to maintain a term vector model that describes the areas of interest of some user. This term vector is fed to the collaborative component to refine predictions. The main advantage of the meta-level method especially for the pair content-based/collaborative is that the learned model is a representation of the interest of a user and collaborative filtering performs better with this representation than with the whole raw data.

3.2.4.8 Summary of main hybridization methods

Table 3 Hybridization Methods

Hybridization Method	Summary
Weighted	Recommendations based in both mechanism
Switching	Switch methods according to some criterion
Mixed	Provides two recommendations at the same time
Feature Combination	Collaborative data is treated as additional information for content-based
Cascade	Use one method to produce recommendations and another to refine
Feature Augmentation	One method produce a rating that is the input of the second method
Meta-Level	One method produces a model that is the input of the second method

3.2.4.9 Advantages and disadvantages of Hybridization

Hybridization mitigates some problems of individual recommendation techniques. However the pair content-based/collaborative always suffers the start-up problem since both approaches need certain amount of knowledge in their knowledge bases to provide reliable predictions. However, hybridization may enable inferring some information departing from sparse databases and in certain way complete these databases. Several proposed hybridization methods are order insensitive while some others are order sensitive. For example those recommenders built using weighted method, mixed method switching method or feature combination will be order insensitive. Several researchers have compared the performance of hybrid recommenders with pure collaborative filtering and found that hybrid proposals can make better predictions especially when the new user problem happens or when dealing with sparse databases Basu, et al., (1998). Main disadvantage of hybrid systems is the complexity of its implementation Popescul, et al., (2001).

3.3 Dynamic nature of user modelling data

Data involving user-modelling changes constantly, new users may arise and new items may be introduced. Additionally sometimes happens that the behaviour of users changes over time. These fact causes two different but related problems to statistical user modelling techniques, these problems are known as: the sparse data problem and the concept drift problem Albrecht and Zukerman, (2007).

3.3.1 The sparse data problem

The sparse data problem also known as cold data problem refers to scenarios where there is insufficient information to classify users or items Albrecht and Zukerman, (2007). In the case of the sparse data problem, both the collaborative and the content-based approach have difficulty to make predictions about a user whom just a few observations are known. In the case of collaborative filtering, systems cannot make reliable predictions about a user insufficiently observed. In contrast, the content-based approach can make such predictions according to the features of these items. However these are not accurate predictions.

3.3.2 The concept drift problem

Concept drift means that user models may become obsolete or scarce. Attributes and features that characterize a user might change over the time yielding outdated models. In real world concepts are often not stable and changes with time. Typical examples of concepts that change are weather predictions or user preferences or likes. In fact the underlying distribution may change as well, that yields to obsolete models with old data inconsistent with new data. Models need a regular updating Tsymbal, (2004). Therefore, the feature of interest can somehow depends on some "hidden context" not given explicitly. For example a customer buying preferences may depend on availability of alternatives, prices etc. Furthermore, the cause of the change use to be hidden and it is not known beforehand, making predictions more complicated. Changes in the "hidden concept" can provoke more or less radical changes in the target concept Widmer and Kubat, (1996).

Two types of concept drift may happen and are normally distinguished in the literature: sudden concept drift and gradual concept drift. In addition, Stanley, (2003) identifies two different types of gradual concept drift depending on the rate of the changes. Thus, two categories of gradual concept drift are, moderate and slow gradual concept drift. Sometimes hidden changes in context (concept drift) are not only a cause of change of target feature or concept but these changes may provoke changes in the underlying distribution. Changes in the distribution might keep the target concept unalterable but may produce unacceptable error rates in the model (virtual concept drift) forcing to update the model Widmer and Kubat, (1993).

3.4 Efficiency considerations

Efficiency issues referring statistical user models check the ability of each model to handle vast amounts of data during main phases of user modelling process. These stages involve the model building (training), prediction generation (inference) and model maintaining. At least second phase requires real time responses Albrecht and Zukerman, (2007).

3.4.1 Context in recommender systems

Even though, most recommender systems provide recommendations just focusing on the tuple user-item, some additional information might be required to provide reliable recommendations Berkovsky, et al., (2007). Thus, the majority of recommenders deals with users (profiles) and items to produce recommendations but they usually avoid put them into context before achieving any recommendation Adomavicius and Tuzhilin, (2011). In the case of recommending a vacation, the system should be aware of some context before providing the suggestion. In this case the recommendations might depend on the season and the user current location. In fact most recommenders should take context into consideration to provide reliable predictions or to improve the quality of their recommendations. Recent companies have started incorporating contextual information into their recommenders. This is the case of Soucetone [4] a web radio that takes into account current user mood (context) before providing next song suggestion.

3.4.1.1 How to understand context

Context is a multidimensional concept studied in different areas over the years. It has been stated that the ability to reach customers anytime and anywhere, depends not only in the quality of products and services but also in the capacity to take context into consideration Prahalad, (2004). Some researchers studied and analyse until 150 different definitions of context from different fields or areas Bazire and Brézillon, (2005). In order to regulate the idea of concept, several taxonomies of context were introduced Dourish, (2004). Thus, context might belong to two different categories, the representational problem or the interactional problem. On the representational side context must be represented in Software systems. Thus the definition of context might be slightly different, in fact several definitions of context within the representational problem can be found in the literature. Context has been defines as "location and identity of nearby people and objects" Schilit and Theimer, (1994). In addition, context has more recently been defined as "any information that can be used to characterize the situation of entities" Dey, et al., (2001). On the other hand, context on the interactional side the context is not only a representation but also what defines a user behaviour.

Even though, several areas have studied context, most important areas related to recommender systems studying the idea of context are data mining, e-commerce, databases, information retrieval, ubiquitous computing and mobile computing Palmisano, et al., (2008). A short review of what is context and how it has been defined will be provided. Regarding data mining, context has been considered as those events or facts that changes customer preferences or likes Berry and Linoff, (1997). Examples proposed of context are finding a new job, a marriage or a divorce. These facts help data mining techniques to focus only in relevant results depending on the context. On the side of E-commerce, context was studied by Palmisano, et al., (2008), who creates different user's profile depending on the behaviour of the user in every circumstance. For example a user might buy from the same site a book for reading or a gift, which makes two different intentions and behaviours. Separating those behaviours in different profiles provides an advantage because it produces a more comprehensive model for different e-commerce applications. The importance of context in e-commerce recommenders has been proved and it has been stated that taking into account context helps to increase the quality of recommendations Adomavicius, et al., (2005). In addition, it has been demonstrated that including dimensions such as time, companion and weather improves the quality and the accuracy of recommendations Oku, et al., (2006).

According to the initial definition of context in ubiquitous computing, context represents the location of the user, the people around the user and main items around the user together with some changes in these components Schilit and Theimer, (1994). This initial definition has been refined over the years including new elements that should take part of context. Examples of these elements are, date, season and temperature Brown, et al., (1997), the interests of the user Ryan,

[4] See: http://www.sourcetone.com/ (2012-08-14)

et al., (1997) and the user's emotional state Dey, et al., (2001). Finally two different approaches, first one is supported by those who relates context to the user Dey, et al., (2001) while second one focuses on relationships between the user and the application Rodden, et al., (1998). In addition, other techniques for context-aware systems have been discussed. For example, hybrid techniques for mobile applications Ricci and Nguyen, (2006) Woerndl, (2007) where several recommenders under the content-based approach and under the collaborative approach (considering context) are available to users to decide the most interesting applications for them. Furthermore, graphical models were introduced, these models enables obtaining the context considering visual information such as images or videos. This approach enriches the context with information non-considered up to this moment Boutemedjet and Ziou, (2008). Finally, all these contextual information becomes essential to provide reliable recommendations to mobile customers Schiller and Voisard, (2004).

Contextual information also becomes important to databases where different queries might provide different answers depending on the context. In fact, a SQL extension was proposed in order to consider not only user's preferences but also contextual information Stefanidis, et al., (2007). In addition, Agrawal, et al., (2006) also incorporate context and user preferences to query language in order to provide suitable answers to contextual queries. Finally, system architecture of a Context and Preference Aware Location-based Database Server (CareDB) was introduced. It personalized content to users depending on its surrounding context. An example of a service that CareDB provides to users is a restaurant finder, in this case CareDB not only takes into account the user location but also the user surrounding context such as user dietary restrictions, user preferences and even road traffic Mokbel, and Levandoski, (2009). It has been proved that combining several location methods such as (Google maps [5], Microsoft MapPoint [6] and Yahoo maps [7]) has resulted in the realization of location-based services for commercial purposes and research prototypes Güting, et al., (2005), Mokbel, et al., (2004). Thus, location services enables providing new services to customers depending on their location. In the case of databases, the query "Where is my nearest restaurant" requires location services to provide reliable responses.

Even though, information retrieval mechanisms often ignored contextual information Akrivas, et al., (2002), it has been proved that contextual information helps information retrieval systems to produce recommendations Jones, et al., (2005). In fact the quality of any information retrieval system resides on its ability to perform context-based retrieval and return context-relevant results Sieg, et al., (2007). Context within information retrieval is considered as the collection of topics close to the search terms. Finally the idea of context into Web services was introduced and well accepted since currently most Web services are context-aware Maamar, et al., (2006).

---

[5] See: https://maps.google.com/maps?hl=en (2012-08-14)

[6] See: http://www.microsoft.com/mappoint/en-us/home.aspx (2012-08-14)

[7] See: http://maps.yahoo.com/ (2012-08-14)

3.4.1.2 Contextual information representation

As previously mentioned, recommendation in its simplest formulation involves estimating ratings for items that a user has not seen before. In this case the rating function R is defined as follows:

R: User x Item → Rating

Actually the R function is estimated for the pairs (user, item) that have not been rated by the users. After that, the recommender might suggest the top-rated item or the N top-rated items. However, context-aware systems take into consideration additional contextual information and the rating function for these systems is defined as:

R: User x Item x Context → Rating

This new formulation is based on the user and items space but also takes into account some additional information such as locations, time, etc. that forms the contextual information. Context is a complex feature that can take many different forms and many different representations, however the most accepted way to represent context is that one based on the hierarchical structure of contextual information that represents context using trees of contextual information Palmisano, et al., (2008). Even with this hierarchical representation, two different approaches have been adopted over the years; first one states that context is a structure that not changes significantly over the time Dourish, (2004). Second approach is called the interactional approach and assumes that context is always changing over the time Anand and Mobasher, (2007).

3.4.1.3 Obtaining contextual information

There are three different ways to obtain contextual information, explicitly, implicitly and inferring Adomavicius and Tuzhilin, (2011). Explicitly means by asking direct questions to relevant people and other sources of contextual information. For instance, a web site might obtain contextual information by asking people to fill in a form. Implicit context is obtained from the environment such as location or time that may be detected by a mobile phone. Therefore, implicit information does not need any interaction with the users, the source of contextual information is directly accessed and information is extracted. Finally, contextual information may be obtained inferring context using statistical or data mining methods. To obtain this type of contextual information a predictive classifier is required, most commonly used is Naïve Bayes. Finally contextual information may be hidden in the data and it can be used implicitly Anand and Mobasher, (2007).

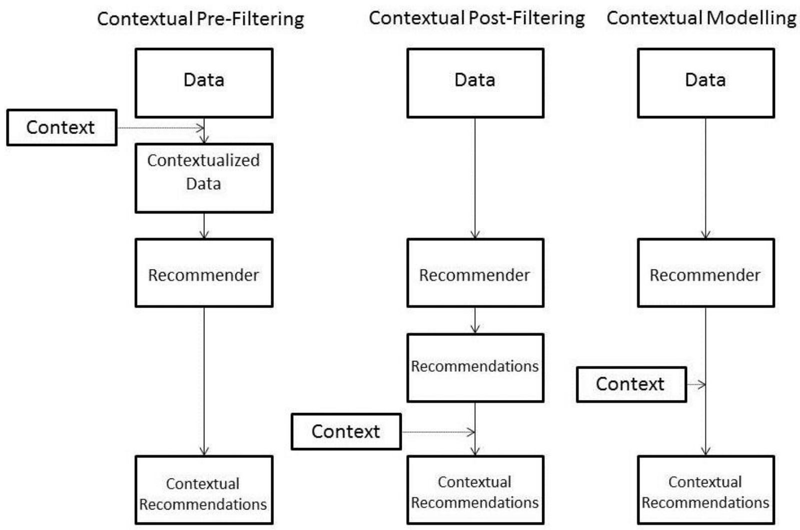
3.4.1.4 Incorporating context to recommender systems

The fact of incorporating contextual information to recommender system can be traced back to the idea that this type of information can lead recommenders to better predictions Herlocker and Konstan, (2001). For example when recommending movies to a child the recommender might perform better if it knows the movies that the children has watched before and which ones he likes. Even though, this approach is under the classical R formulation (R: User x Item → Rating), it leads to the idea that additional information can be incorporated to the standard filtering approach. In fact, the use of interest topics (likes) in recommender systems was used to build contextual profiles for recommender systems Ziegler, et al., (2005). Two main approaches have been used to incorporate contextual information to recommender systems. First approach uses contextual information obtained directly from the user, or directly from the environment (specify mood, location) to query some repository and present the best matching item. Main applications of this approach are mobile recommenders and tourist recommenders Cena, et al., (2006) Van Setten, et al., (2004). Second approach to incorporate contextual information to recommenders involves modelling and learning user preferences by observing user's interactions with systems and other users Panniello, et al., (2009) Yu, et al., (2006). Finally both approaches have been combined for example to produce news recommendations Cantador and Castells, (2009).

Hereafter, an overview of second approach can be found, this second approach is more recent than the first one and has better acceptation to build context-aware recommenders. Three different methodologies have been used to construct recommenders under this approach. These approaches are, contextual pre-filtering, contextual post-filtering and contextual modelling.

Contextual pre-filtering takes into consideration contextual information during the process of recommending items. Therefore the acquisition of contextual information is done previously to the recommendation process Adomavicius, et al., (2005). Contextual post-filtering initially ignores the context information and produce recommendations without considering the context. Afterwards, the set of recommendations is fine-tuned to get adapted to contextual information Panniello, et al., (2009). Finally, contextual modelling involves using contextual information as part of rating estimation. Thus, contextual modelling uses context directly in the recommendation function and to predict or estimate the ratings of each item. In fact, this approach is the only one that truly uses the context-aware function (R: User x Item x Context → Rating). Figure 33 depicts main ways to handle context when performing recommendations.

Figure 33 Context on Recommendations



Several examples combining these approaches can be found in the literature together with how these combination can lead to better performance Burke, (2007) Koren, (2008). In addition, several combination methods have been used. First one involves combining several models of the same type Adomavicius, et al., (2005). Second one involves combining two different approaches; most common combination involves pre-filtering and post-filtering methods.

3.4.2 Statistical user models evaluation

The performance of a recommender must be measured according to an evaluation process. The type of metrics user to evaluate a user model depends on the collaborative filtering. Metrics to evaluate recommendation systems can be classified as follows: predictive accuracy metrics, classification accuracy metrics and rank accuracy metrics Herlocker, et al., (2004).

3.4.2.1 Evaluation issues

The evaluation of any recommender system is an inherently difficult task due to the following reasons. First one refers to the data set different algorithms may perform better or worse depending on the data set. Most collaborative systems are designed for a specific data set or maybe for a concrete data set structure. For example MovieLens Miller, et al., (2003) has designed for a dataset with 65.000 users and 5.000 movies. In the case of MovieLens dataset, there are much more users than items and this system might be inappropriate for any domain with more items than users. Similar issues occur with other properties of the datasets. Second reason concerns evaluation goals, most evaluations differ in their goals. For example most evaluations focuses on measure the quality of recommendations when it could be more valuable to measure how often the system leads to wrong choices. For example some works may measure large errors between the predicted and the proper item Shardanand and Andmaes, (1995), other work may suggest that properties different from accuracy might have higher effect on users satisfaction and performance. Finally a significant challenge happens at the time of deciding a set of comparable metrics. It seems that new systems perform better when comparing with a similar older system, but several authors identify that when both systems are tuned to its maximum, results are similar. For example some authors suggest that users provide inconsistent ratings when asked to rate an item at different times Hill, et al., (1995). They propose that an algorithm cannot be more accurate than the variance in user’s rating.

3.4.2.2 Data sets

Data sets main properties can be divided into three different categories: domain features, inherent features and sample features Herlocker, et al., (2004). Each of these three categories has some features that must be analysed. Domain features reflect the nature of the content being recommender independently of the system itself. Main domain features are.

- Topic of recommendations.
- The user tasks supported by the system
- Quality need
- The granularity of user preferences

Most recommenders are built for a particular domain; in fact most typical domains are related to entertainment (movies, music, etc). However, a particular topic might have different contexts, for example movie recommenders may operate totally on the web or may operate as a part of a digital video recorder. Probably the most important feature is the quality needed for recommendations, most users were generally happy if recommendation includes just an item they were not familiar with it. This approach matches the conventional recommendations but it is almost useless for some recommendation domains, for example the supermarket domain where new items cannot be discovered and this approach cannot be used to recommend bananas or eggs. Another important domain feature refers to user granularity, which is how many levels of user preferences are. For example in binary domains users just rate items as good or bad items. In that case the distinction among good items is not necessary as the distinction among bad items. However, user granularity may be different than ratings granularity, for example a user might rate an item from 0 to 10 but obtain recommendations that only care if the item is good or bad. In general evaluating an algorithm with significantly different domains features can be a mistake.

Inherent features reflect the core nature of the recommender system. Main inherent features are.

- Whether rating are implicit or explicit
- Scale of ratings
- Timestamps
- Context information

Explicit ratings are entered by the user, while implicit ratings are inferred departing from explicit ratings. The simplest explicit scale is the unary-liked where all that is known is whether the user likes or purchased an item or not. The term unary is preferred to binary because the lack of purchase of item X does not mean that the user does not like the item. Explicit ratings scales usually have 5-point, 7-point or 100-point scales. Most recommenders uses a single rating dimension, however systems where users can enter several ratings for a single item are becoming important. Timestamps are a property especially important for those users that are prone to change its preferences according to their interaction with other items. Finally the presence of demographic information could improve recommendations, however some researchers identify that most answers related to demographic information are false due to the suspect of marketing questions Herlocker, et al., (2004).

Sample features refer to data distribution and it can usually be manipulated by selecting the appropriate subset. Sample features include most of the statistical properties considered during the evaluation. Main sample features to consider are.

- Sample distribution
- Density of ratings

Sample distribution refers to the number of users and number of items and the equivalence between these features. In addition, density of ratings means the average of items that have been rated. Some dataset might have uneven distributions and density of ratings may be manipulated including or excluding items. Each of these statistical features has great impact on recommender results. The relationship between number of users and number of items determines whether it is easier or not to discover correlations among users and among items. Ratings distribution also affects system performance, dataset with dense sub region of ratings and sparse sub region of ratings might use the inferences over the dense region to recommend items of the sparse region.

For example, the system Jester, Goldberg, et al., (2001) was designed to ask all users to rate the same jokes, thus the system makes a dense section of ratings. Afterwards, the system employs inferences over the dense section to recommend jokes in a sparse section. On the other hand, systems using more even dataset will be more challenged to cope with the sparse sections unless they incorporate dimensionality reduction techniques Billsus and Pazzani, (1998).

3.4.2.2.1 Main datasets

Over the years some dataset have been used to evaluate most recommender systems, more widely used datasets are EachMovie dataset (http://research.compaq.com/SRC/eachmovie/), Jester dataset Goldberg, et al., (2001) and MovieLens dataset. EachMovie dataset contains more than 2.8 million ratings from 70000 users and it has been used in several research projects to analyse how to predict user ratings. For example analysis algorithms have been tested using EachMovie dataset Canny, (2002) or algorithms to compute network values Domingos and Richardson, (2003). MovieLens dataset has not been totally released but a sub-dataset of 100000 ratings and 1 million users was released and used mainly for cold-start recommendations Schein, et al., (2001). Finally Jester dataset has been used for several researchers to evaluate their recommender systems. Note that this dataset is totally different from the EachMovie dataset and the MovieLens dataset since it suffers from much more biases than the previously mentioned and it provides a dense region of ratings. Most publications on recommender systems use these three dataset. Researchers do not have resources to build their own dataset or to collect data to create the dataset, however they are forced to use the few existing datasets.

3.4.2.3 Evaluation metrics

Main problem to select one or more metrics to evaluate collaborative filtering algorithms is that there is a lack of standardization. Researches must face some questions at the time of selecting metrics for evaluation. Are the results with this metric comparable to other works in the same field? How sensitive is a metric to detect differences if there are? According to Herlocker, et al., (2004) answers to these questions have not been totally addressed. In addition, the problem of standardized metrics is still growing since researchers introduce new metrics when evaluating their recommender system. It is almost certain that compare results from one publication to another publication is pretty difficult according to the diversity of evaluation metrics user for collaborative filtering.

3.4.2.3.1 Most commonly used metrics

Recommender systems accuracy has been subject of discussion since the middle of the nineteen's Resnick, et al., (1994). As mentioned above, evaluations of recommender systems used many different metrics (see 2.3.4.3). Predictive accuracy metrics are based on true user ratings and compare these ratings with system predictions to measure the distance between true ratings and predicted ratings. Predictive accuracy metrics are very important for those recommenders that displays predicted ratings to users. MovieLens predicts the number of stars that a user will give to each movie and displays the prediction to the user. Main problem of these metrics is that even if the systems rank properly the item it could fail due to the predicted rating. That is main reason why most systems just display a list of ranked items but they do not display the predicted ratings.

In any case, each researcher interested on using predictive accuracy metrics is restricted to a measurement of the distance between the true rating and the predicted rating, for example the mean absolute error. Mean absolute error (MAE) calculates the absolute deviation between true rating and predicted rating Willmott and Matsuura, (2005). It has been used to evaluate several recommenders Breese, et al., (1998). Mean absolute error formula corresponds with the following equation.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|Y_j - Y'_j\right|$$

Main advantages of mean absolute error are the capacity to measure the accuracy of predictions, its simplicity and easy to be understood and finally that mean absolute error provides good properties to measure the difference between the errors of two systems. These properties related to mean absolute error are, mean squared error; root mean squared error and normalized mean absolute error Goldberg, et al., (2001).

Classification metrics are suitable for those recommenders intended to classify items as good or bad items. These metrics measures the ability of some recommender to make correct or incorrect decisions. Thus classification metrics are appropriate for binary classification. On the other hand, classification metrics may be challenged by the data sparsity problem. To avoid this problem during evaluations, main approaches are. First one suggests removing those items that have not been rated by users Herlocker, et al., (2002). Second approach to avoid sparsity problem is to assume default ratings for those items that have not been rated by users (commonly negative ratings) Breese, et al., (1998). On the side of metrics borrowed from the area of information retrieval, precision and recall are used in information retrieval Cleverdon, (1968). Even though, these measures were used in information retrieval, they have been used for evaluating recommender systems Bassu, et al., (1998). Precision and recall are computed in a binary scale where the item is classified as relevant or non-relevant. Some systems used larger than binary scales but transform these scales into a binary classification. For example MovieLens uses a rating scale from 1 to 5 but it considers ratings higher than 3 as relevant and the rest as non-relevant Dahlen, et al., (1998). Precision is defined as the ratio of relevant items retrieved from the number of items selected. Following equation displays precision definition.

$$Precision = \frac{|\{Relevant\ documents\} \cap \{Retrieved\ documents\}|}{|\{Retrieved\ documents\}|}$$

Therefore, precision measures the probability that a retrieved item was relevant. On the other hand, recall is the ratio between relevant items selected and the total of relevant items. Recall is defined as follows.

$$Recall = \frac{|\{Relevant\ documents\} \cap \{Retrieved\ documents\}|}{|\{Total\ Relevant\ documents\}|}$$

However, both precision and recall depend on the idea of relevant and non-relevant. In addition, the meaning of relevance has been discussed over the years in the area of information retrieval Harter, (1996). In fact the area of Information Retrieval employs the objective meaning of relevance where something is relevant with respect to a query or a document and a set of specialist can decide whether it is or it is not relevant. In the field of recommender systems this meaning of relevance does not make sense since recommenders produce recommendations according to a certain probability. In fact, the user is the only person who can determine whether the item is relevant or not. Therefore in the area of recommender systems, relevance must be considered in its subjective meaning. Main problem of considering relevance in its subject meaning is that for some user an item rated with 3 might be relevant but for another user it might be non-relevant. This reason makes that researchers had to put much effort researching on multipoint scales. Thus, ROC curve is an alternative to precision and recall measures that was introduced with the name of "relative operative characteristic" Swets, (1969). ROC curve measures the ability of a recommender system to differentiate between relevant items and noise. Thus, ROC curve is a two-dimensional representation of recommenders or classifiers performance. Usually when some ROC curve is higher than another one means that this system performs better than the one corresponding with the lower curve. The area under the ROC curve measures the system performance.

Regarding measures that compute the error rate of a recommender, there exist Ad hoc measures. Ad hoc measures define the error rate as the number of incorrect predictions over the total of predictions. However with just a few recommendations the error rate can only be measured experimentally. In addition Ad hoc measures also focus on larger errors in recommendations, for example errors of three or more points on a five-point scale. Ad hoc measures might be interesting for preliminary evaluations but these measures have not been used still for final evaluations Herlocker, et al., (2002).

Rank accuracy metrics are suitable for those recommenders intended to provide an ordered list of relevant items to the final user. This metrics are used when recommendations are non-binary. Even thought that a recommender provides a list of ten relevant items the rank metric might not yield the highest value because the most relevant item may not be in the first position of the list. Thus, the purpose is not only to provide a set of relevant items but also providing a well-ordered list. In addition, some recommenders distinguish between a relevant item and the most relevant item, in such domains the list of recommended items might contain relevant items but these items may not be the most relevant ones.

3.4.2.3.1.1 Other metrics

Several other metrics have been used to evaluate user models. For instance the prediction correlation where two variables are correlated if these variables are dependent that is the variance in one variable depends on the variance on the other. Most widely spread correlation measures are Pearson product-moment correlation Pearson, (1896), Spearman rank correlation coefficient Spearman, (1904) and Kendall Tau correlation coefficient Kendall, (1938). Despite the simplicity of all these correlations, their uses have been restricted to just a few projects. However, correlation metrics have some advantages like that they are well understood by researcher's community, or that they provide a single measure for the whole system. Main disadvantage of correlation metrics is about interchanges. For example an interchange between recommendations 1 and 2 will have same impact in metric outcome that interchanges between recommendations 100 and 101 since both interchanges have the same distance in positions.

Another commonly used metric is the half-life metric. It was designed for those processes that need a list of ranked elements but the final user is not supposed to go into the details of the list, main example of this process are web browsers Breese, et at., (1988) Heckerman, et al., (2000). This metric measures the utility of the ranked list for the user. The metric measures the likelihood that the user views every element according to its position in the list. First element must have greater likelihood and successive elements will have lower probabilities according to a decay function. The strength of the decay is defined by a half-life parameter. Therefore to get higher values of the metric first elements in the list must be those with higher rating by the user. Main disadvantage of half-life metric concerns its parameterization; it might vary from one researcher to another researcher making evaluations almost non-comparable. On the other hand, the NDPM metric was introduced in 1995 by Yao, (1995) and it was mainly used to evaluate the recommender FAB Balabanovic and Shoham, (1997). NDPM consists of a normalized distance-based performance measure. Main advantage of NDPM is that it is comparable among different datasets since it is normalized. However, as happens with correlation metrics NDMP suffers the problem of ratings interchanges. In addition, NDMP just evaluate orders but it is not appropriate to evaluate ratings values.

Some researchers accomplished a comparative among evaluation metrics and some interesting conclusions were extracted from this comparison Herlocker, et al., (2002). Any metric computed per individual user and averaged provides different results than the same metric computed overall. Mean average metric provides same order than correlation metrics computed per user and averaged. Pearson correlation metric provides almost same results than rank correlation metrics. ROC area metrics perform similar to other metrics when computed overall. Finally the study concludes that some metrics are more appropriate depending on the recommender and the task of evaluation. Thus some risk of evaluating a recommender using an inappropriate metrics arises. In this case results provided by the evaluation metric will not be reliable and comparable.

3.4.2.3.2 More Recent Metrics

Recent studies about recommender's evaluations shows that previous metrics based on accuracy are not enough to determine the effectiveness of the system and the satisfaction of the final users. In fact recommenders evaluations based on accuracy might yield a high value of the metrics since evaluations may be based on easy to predict items. In addition some evaluations of recommenders provide deceitful results even when evaluating with the proper metric. For example a recommender might be intended to provide a large ordered set of items to final user while the user is expecting just a few highly useful recommendations. Thus, the user task also affects in evaluation and might disturb evaluation results Turpin and Hersh, (2001). Thus, evaluations metrics have been moving to measure not only accuracy but also suitability of recommendations. Most important new trend metrics are detailed below.

First and one of the most used metrics is called coverage. There exist two different forms of coverage concerning recommender systems. First and most common form of coverage is known as prediction coverage measures the number of items that a recommender is able to recommend. Second form of coverage is known as catalog coverage and it is defined as the percentage of items that the recommender is able to suggest from the available items Zanker, et al., (2007). System with lower coverage will probably be less valuable for users since they are limited in choices. Coverage has been employed by many researchers to measure the ability of their recommenders to provide reliable predictions Good, et al., (1999). On the other hand, collaborative systems can be evaluated using their learning capability. The prediction capabilities of these recommenders should improve as the amount of data available increases. However there is no general rule for statistical algorithms, some might need just a few points to start producing acceptable recommendation while other may need more representative data points. The amount of data-points needed by a recommender is known as the learning rate. There are three main learning rates concerning recommender systems, overall learning rate, item learning rate and user learning rate Schein, et al., (2001). Overall learning rate is a measure of quality in recommendations computed over the total number of items or users. Item learning is measured for a single item according to the number of ratings that the item has. Finally user-learning rate is calculated as a function of the items that the user has rated. Last but not least, innovation, serendipity and confidence can be tested to evaluate a recommender system. Most recommender system might provide acceptable predictions based on obvious items. For example the recommendation "buy apples" in a supermarket probably is a highly accurate recommendation but it lack in usefulness. Thus, obvious recommendations might improve the accuracy of the system but they do not make any effect in user satisfaction. However, it has been found that obvious recommendations might be suitable for new users even when these recommendations do not provide any new information. Main reason is about system credibility, these recommendations about items familiar to the user increases user confidence in the system Swearingen and Sinha, (2001).

Two new dimensions needed to evaluate recommender systems were identified, first one is innovation and second one is serendipity Herlocker, et al., (2004). Innovation means that the recommender is able to suggest items related to other items known to the user. On the other hand, serendipity is related with the ability of some recommender to suggest items that are not even related to other items known by the user. For example a recommender that suggests a new film of some director known to the user provides innovative recommendations, while a recommender that suggests a completely new film by a new director provides serendipitous recommendations. A serendipitous recommendation will be also an innovative recommendation. In addition, the distinction between innovation and serendipity is important because serendipity is an intrinsic feature of collaborative filtering systems and content-based systems cannot provide this feature. Best metric to measure serendipity of a recommender system should check how serendipitous recommendations increase user's interest for these suggestions.

Finally confidence refers to how accurate are recommendations that recommenders suggest. In fact most users must face the problem of how to interpret recommendations when no measure of confidence is given by the system. In addition, most recommenders systems are based on the idea that a user is more likely to prefer an item rated five stars in a five stars scale than an item rated four stars in the same scale. However Herlocker, et al., (2004) discovered that this assumption is often false according to the problem of data sparsity. High predictions might be suggested just because the small amount of data about these items. Some researchers have been done about how to display the confidence of a recommendation; main conclusion is that the way that confidence is presented to user makes significance differences in user's decisions. In fact, best confidence displays are much better than no display. However, worse displays have a bad impact on decisions Herlocker, et al., (2000). Thus, it can be stated that recommenders that do not include confidence measure are prone to provide poorer decision-making to users.

3.4.2.3.3 Evaluation remarks

Even though, all metrics discussed above affect in evaluation, there are some other issues that must be considered during the evaluation of a recommender system. In fact the whole space of evaluation is larger than the space of the previously analysed metrics. Most important dimensions during evaluations were identified and analysed by Herlocker, et al., (2004).

Ask versus observe: First dimension refers to how the system gains insight into the user. Explicitly (ask) or implicitly (observe), first method usually employs interviews while second one monitors user behaviour and subjects it to several analyses.

Laboratory studies versus field studies: Laboratory studies test the system under certain well-controlled conditions. However field studies might show new real contexts or new real patterns that developers may not have considered.

Outcome versus process: Accuracy might be the most important metric since the system perspective. In fact suitable metrics must be designed taking into account how to reach a successful outcome Newman, (1997). According to user's perspective these metrics must be defined considering their particular evaluation tasks.

Short-term versus long-term: Some issues might not arise in short-term evaluations (Lab studies) while they may appear in long-term experiments.

Several researchers analyse these evaluation dimensions in its studies Cosley, et al., (2003) McDonald, (2001). Mainly they accomplished several laboratory studies that explicitly gather information from users. Other researches achieve quite similar experiments but using both explicit and implicit information in their studies Amento, et al., (2003).

Together with these evaluation dimensions, most recommenders provide to users a rating about its recommendations. In fact, Cosley, et al. (2003) achieved a complete study to determine how predicted rating can affect the current rating given by the user. This study found that inaccurate predictions reduced user satisfaction. Therefore, there is some evidence that users are sensitive to prediction's accuracy. However, there still more than accuracy, users should rely on the recommender system and recommendations of familiar items aids in this process. Furthermore, information about why some item has been recommended makes users gaining confidence in the system. Supporting information about recommendations might help users to understand the reason of the suggestion and to trust the recommender system. For example recommending a movie by suggesting just its title may not be enough to convince the user to watch the movie. On the other hand, suggesting the same film but providing supporting information about its director, some reviews or some trailers should be more appropriate to convince users. Regarding the metrics, they can vary from one domain to another domain and from one evaluation task to another evaluation task Amento, et al., (1999).

3.4.3 Improvements of statistical recommenders

Previous approaches to statistical recommenders can be upgraded incorporating new features or components like better understanding of users and items, contextual information, multi-criteria ratings or more flexible recommendations.

3.4.3.1 Better understanding of users

Some recommendation systems do not take full advantage of the information about the user and its previous sessions. In fact these systems produce predictions based on restricted understanding of users Konstan, et al., (1998) Adomavicius and Tuzhilin, (2001 a). Main examples are classic collaborative filtering algorithms that just use ratings information to produce predictions instead of using users and items profiles to yield these predictions Resnick, et al., (1994) Hill, et al., (1995). Even though these systems do not use user's profiles and that there have been an improvement in profiling over the years, profiles in recommender systems tend to be quite simple Pennock and Horvitz, (1999). Thus, more advanced profiling techniques based on data mining techniques, sequences and signatures should be use to improve the performance and exploit the power of user's profiles.

3.4.3.1.1 Data mining profiling techniques

There has been a great interest over the years in one-to-one marketing applications Peppers and Rogers, (1993) and in recommender systems able to suggest individual user's items and services of their interest Kautz, (1998), Soboroff, et al., (1999). Thus, personalized profiles must contain as much as possible information about the user. These profiles are models containing user's behaviour and preferences; they can be built departing from user's previous sessions and using data mining techniques. However several rules in a profile might be ambiguous or irrelevant, main problem is about how to perform rules validation. Mostly this validation should be done with human expert support. Even with human support some problems may arise due to the amount of rules per user or due to the amount of users. Two main approaches have been adopted to build user's profiles, first one involves constructing profiles departing from demographic and transactional data, these profiles contain important factual information about the user. Main examples of factual information in these profiles are, age, address, incomes, show sizes and certain facts. All these features are extracted from the user transactional data. On the other hand, in the second approach the profiles contain not only factual information but also rules concerning user behaviour. However most profiles built under this approach have rules for groups of users instead of having rules for every individual user Adomavicius and Tuzhilin, (2001 b).

3.4.3.1.1.1 User profile definition

Data in user's profiles has been classified in two different types, factual data and transactional data. Usually factual data is related to who is the user while transactional data is related to what the user does. On the other hand, the profile itself is a set of information describing a user. Main issue concerning user's profiles is about what information should be in every profile. Simple user's profiles include only factual information while more complex profiles relies also on transactional information. Profiles containing transactional information contain a set of rules describing user's behaviours. User's behaviours can be defined by means of conjunctive rules such as association rules Agrawal, et al., (1996) or classification rules Breiman, et al., (1984). Using rules in profiles provides information about user's behaviours. An example of rule describing a user behaviour is "when user X comes to site Y from site Z he usually returns to site Z".

3.4.3.1.1.2 User profile construction

Several data mining algorithms have appeared over the years. Most important algorithms are, Apriori Agrawal, et al., (1996) using association rules, CART Breiman, et al., (1984) using classification rules and C4.5 Quinlan, (1993). Several studies reveal that some rule discovery algorithms produce very large set of rules where many of these rules are almost non-relevant, trivial or false Piatetsky-Shapiro and Matheus, (1994), Padmanabhan and Tuzhilin, (1999). Thus rule validation becomes an important task in order to alleviate the amount of deceitful rules.

3.4.3.1.2 Sequences and signatures in profiling

Several data mining techniques have been used to discover patterns or sequences of events. Main applications of these techniques are discovering sequences in alarms, user interfaces, and crimes committed Mannila, et al., (1997). In addition the interest on sequences discovering has increased over the years Agrawal and Srikant, (1995) Bettini, et al., (1996). One of the most important problems in discovering sequences involves finding episodes Mannila and Toivonen, (1996). Episodes are set of events that happen together in a large sequence of events. Once the episodes have been identified, second step involves finding relationships among the different events in the episode. Such relationships can provide information about a user or an item and improve the quality of its profile. The basic idea is look for small frequent episodes first and then look for larger episodes. In addition, Mannila, et al., (1997) defines a sequence of events and an episode as follows:

A sequence of events consists of several events together with their time of occurrence. Therefore an event can be defined as a set E of event types where an event is a pair (A, t) where $A \in E$ and t is the occurrence time of the event. In addition, the event type (A) might contain several attributes. On the other hand, an episode is defined as a partially ordered collection of event occurring together.

Regarding signatures in profiling, a signature is defined as a set of fields in a data stream. For example a signature for a phone number might be a set of measurable features such as information about the regions where most calls to this number are placed. Therefore signatures can be considered as evolved profiles where transactional information has been included. Main application to extract information or signatures from large data streams is Hancock (Cortes, et al., 2000). Transactional information obtained from data streams can improve the quality of any profile.

3.4.3.2 Best practices in recommendation techniques

Even though, statistics and machine learning areas helps in rating estimation, some other areas of mathematics and computer science might provide better features to develop improved rating estimation Buhmann, (2001) and Nurnberger, (1989). For example, the use of radial basis functions Duchon, (1979) and Schaback and Wendland, (2001) might help to improve ratings estimation. Main advantage of radial basis functions is that they have been studied and analysed over the years. In addition, the use of these functions for practical applications has good acceptance and reputation within the mathematics community. Thus, radial basis functions leads to an encourage alternative to improve recommender systems. Therefore, there are methods different from statistics and machine learning to improve the quality of recommender systems. Main approaches to improve recommendations and users satisfaction are, multidimensional recommendations, multi-criteria ratings, non-intrusiveness and effectiveness in recommendations Adomavicius and Tuzhilin, (2007).

3.4.3.2.1 Multidimensional recommendations

Multidimensional recommendations refer to the ability of recommender systems to take into consideration additional information such as context (see 2.3.4).

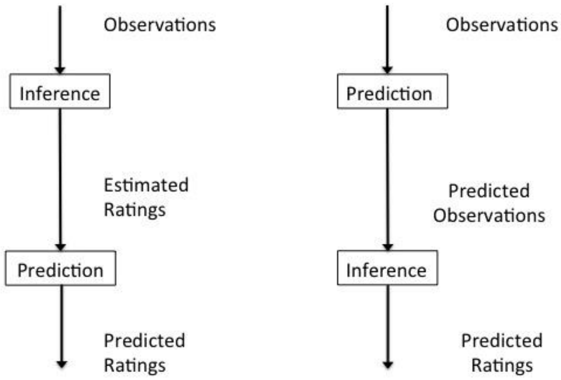
3.4.3.2.2 Multi-Criteria ratings

Even thought, most recommender systems deal with single ratings condition, for some recommenders it may be crucial to consider multi-criteria ratings. For example a recommender for restaurants should take into account not only the food but also the service to perform its suggestions. Main approaches when performing multi-criteria ratings are: Pareto optimal solutions, reducing multi-criteria problems to single criteria problems, optimizing most important criterion.

3.4.3.2.3 Non-intrusiveness

Intrusiveness in recommender systems refers to the degree of involvement of the user with the recommender. In addition, some recommenders are considered intrusive since they require some feedback from the user. On the other hand, some other recommenders are non-intrusive since they use implicit feedback instead of requiring explicit feedback. Recommender systems usually exploit explicit ratings given by users. The idea of providing explicit ratings usually makes difficult to get large populations and to get reliable ratings. In addition, explicit ratings contribute to data sparsity problem. Implicit feedback tries to avoid this bottleneck and provide large populations together with reliable ratings. Three different sources of implicit feedback have been identified, examination, retention and reference Nichols, (1997). Examination means monitoring user interactions to discover repeatable behaviours. Retention involves grouping behaviours that suggest intention of making use of an object. Finally, reference seeks relationships among objects and actions performed just before or after using that object. Two different strategies have been adopted to use implicit ratings. First one employs observations to inference estimated ratings and use these estimated ratings to produce predicted ratings Konstan, et al., (1997). Second approach uses past observations to predict user’s behaviour in response to new information and then the inference stage to produce predicted ratings Stevens, (1993). According to Oard and Kim, (1998), Figure 34 depicts these two strategies.

Figure 34 Strategies for implicit ratings



Even though, both strategies are quite similar, there are some differences. First strategy will characterize some item departing from information observed of other users, while second strategy characterizes items departing from predicted ratings for other users. An example of non-intrusive ratings is the time that some user spends reading an article. Even though, these implicit ratings are useful to improve the quality of predictions, they cannot totally replace explicit ratings provided directly by users due to their inaccuracy Adomavicius and Tuzhilin, (2007). Thus, researchers must address the problem of intrusiveness and decide the amount of explicit ratings asked to users and the amount of implicit feedback that guarantees reliability in their systems. There are different ways to handle the intrusiveness problem, for example defining a set of explicit ratings that the user must provide before using the system. This approach involves some effort on the side of the user. Finally, the problem of selecting data from both the implicit and the explicit side resides on developers.

3.4.3.2.4 Effectiveness

Effectiveness in recommender systems refers to the quality of the metrics used to evaluate recommenders together with the ability to build reliable metrics (see 2.3.5.3).

3.5 Highly-Related work and summary

There are not many proposals of the same aim and nature than the here presented, but during last years this research area has experimented a certain growth in the application cases, which undoubtedly should encourage research and subsequently the appearance of new approaches. Those applications usually show very small deviations to common and largely applied classic solutions, which have proven both reliability and versatility. Though, there is still much space for improvement to explore. In Table 4, main features of proposals and systems highly related to this research are briefed.

Table 4 Comparison of highly-related systems

	Model approach	Collaborative approach	Context management	Section
(Konstan et al., 97) GroupLens	Collabo-rative	Memory-Based	Contextual Pre-Filtering	3.2
Fab (97)	Hybrid	Memory and Model Based	Contextual Pre-Filtering	3.1
(Billsus & Pazzani, 99) Daily Learner	Content Based	Memory-Based	Any	3.1
P-Tango (99)	Hybrid	Memory-Based	Any	3.3
(Goldberg et al, 01) Eigentaste	Hybrid	Model-Based	Any	3.3
(Ricci & Nguyen, 06) Mobyrek	Hybrid	Model-Based	Contextual Pre-Filtering	3.3
present proposal	Hybrid	Memory and Model Based	Contextual Pre-Filtering and Modelling	4

Besides, experts and researches in the area have identified some obstacles and risks when applying those solutions to specific cases. Such hurdles force to evolve the technology in order to avoid falling into fallacies and model degenerations, which could be produced in practice, endangering the endeavour success. In Table 5, main problems found in the state of the art are compared with most acclaimed approaches, in order to spot which systems are capable to deal with each of those problems. Finally, the here presented proposal is added to that table to state the research goals and characterize the strengths and weaknesses of this approach, which is the core of this work.

Table 5 State of the Art summary

	Over Specialization	New User Problem	Data Sparsity	Scalability	Synonymy	Grey Sheep
(Konstan, et al., 1997) GroupLens	×	×	×	×	×	×
Fab(1997)	×	≈	×	×	×	×
(Billsus and Pazzani 1999) Daily Learner	×	≈	✓	×	✓	×
P-Tango (1999)	≈	×	×	×	×	×
(Goldberg, et al., 2001) Eigentaste	×	≈	✓	×	×	×
present proposal	×	≈	✓	✓	×	×

3.6 Concluding remarks

Summarizing, both content-based and collaborative filtering have to deal with several problems such as the new user problem, the over-specialization problem, the data sparsity problem, the grey sheep problem and scalability problems including inefficient response times. It can be stated that hybrid approaches combining content-based and collaborative filtering are an encouraging solution to overcome all this challenges. Since both content-based and collaborative filtering present several advantages and several drawbacks, the combination of both systems usually overcomes most of the drawbacks of each technique. Actually, some of these problems remains being main challenges in the area of user modelling as identified by Albretch and Zukerman, (2007) as long as efficiency considerations and the use of dynamic user features.

The last big challenge of user modelling and consequently of recommender systems is in their evaluation. As previously stated the lack of proper mechanisms and evaluation frameworks makes the evaluation of these prototypes a hard task. The lack of an adequate evaluation framework not only refers to the lack of evaluation mechanisms but also to the lack of methodologies, metrics and experiments properly adapted to the problem of user modelling. Thus, it can be stated that to date there have been no published attempts to synthesize what is known about the evaluation of user models and recommender systems, nor to systematically understand the implications of evaluating these systems for different tasks and contexts.

Finally, it is remarkable the need of a new approach on the area of user modelling addressing main challenges of the area like scalability, response times, domain independence and knowledge loss. On the side of evaluation, the lack of evaluation frameworks for user modelling turns out into the need of adapting mechanisms and built a complete Benchmark to make user models results comparable and to provide the area with standard metrics and methodologies.

4 Proposal

Main goals of this research were initially described in the *Introduction section* of this document. Goals can be classified in two main categories: those goals related to the construction of a new user model and those goals related to the problem of evaluating user models. Even though there are a clear connection between both categories, this distinction clarifies what is being tried to be achieved with every of further decisions taken.

Therefore, first category involves building a new innovative user model that overcomes most of the limitations of traditional user modelling. As previously stated, traditional user modelling has several restrictions such as: domain dependent models or models with knowledge loss. Thus, the new approach must deal with knowledge loss and be able to perform in different domains providing reliable predictions. The idea of building a domain independent user model where the whole model can be reused when changing the data source clearly provides advantages regarding those models that have been specifically designed for one particular domain. However, changing the domain while keeping the same model might lead to performance drop. Therefore, the proposed model must not only be able to perform in such different domains but also be able to provide at least as good performance as traditional user models. On the other hand, the problem of knowledge loss usually affects traditional models reducing their reliability. This problem has a direct impact in the performance of most models. In fact the idea of knowledge loss derives from the scalability problem (Section 3.2.2.2). Over time user models have more and more sessions gaining knowledge from the users during those sessions. As sessions passed, the size of the knowledge base grows over some bounds. In that case, the most common solution is merging knowledge with the consequent loss of information. However, that merge not only keeps the size of the knowledge base under certain limits but also keeps the efficiency of the model. Another problem related with merging the information is called certainty degradation (Section 3.2.2.1), after several merges, the certainty of the resultant groups trends to be much lower than the certainty of other groups that have experience less merges. Even tough, the solution of merging knowledge has provide reasonable results over the years, this thesis tries to provide a new model improving the results of traditional user modelling by solving the scalability problem and consequently the problem of knowledge loss.

Before going into details of the new proposal, some general concepts of every user model are detailed. Those concepts and definitions help to have better insight about the problem of knowledge loss and about the scalability problem.

The first concept that must be described is the notion of user description (UD) (1); a user description stores each single information atom that defines a user. Each of those atoms represents one feature of the user. Thus, a user description is defined extensively as a set of information atoms represented by tuples of the type {feature, value, sign, certainty}. Formally, a user description is defined as follows:

$$\text{Let UD be} \subset \{(c, v, s, z): c \text{ is a user feature}, v \in \text{domain}(c), s \in \{-1,1\}, z \in (0,1]\} \quad (1)$$

It is remarkable that each tuple in a given user description has a direct impact characterizing that user: the value applies (or not, depending upon the sign) to the user regarding the feature, observing that this piece of information has a given certainty, that is, a measure of the confidence of being certain when applying this statement. The term certainty is preferred to probability since there are diverse sources of knowledge including the user's fuzzy assertions through his/her interventions. The certainty must be higher than zero, given that value zero is considered as 'no information', and any tuple with such certainty will be omitted. Any UD may lack tuples for a given feature (which is equivalent to including a tuple with certainty zero for each value and sign in its domain) or include several rows with the same feature but different values.

The UD includes no identifier, so this description initially defined to stand for a user may actually represent several similar users. Consequently, a stereotype (henceforth, group of users) is also a UD with the addition of a value for its 'population', indicating the number of sessions held to acquire this group. Notice that the term *session* is here preferred to the term *user*, since there are no identifiers and therefore all those similar UD (or part of them) could have been acquired through several interactions (sessions) held with the same user. Formally, the definition of a group of users is (2):

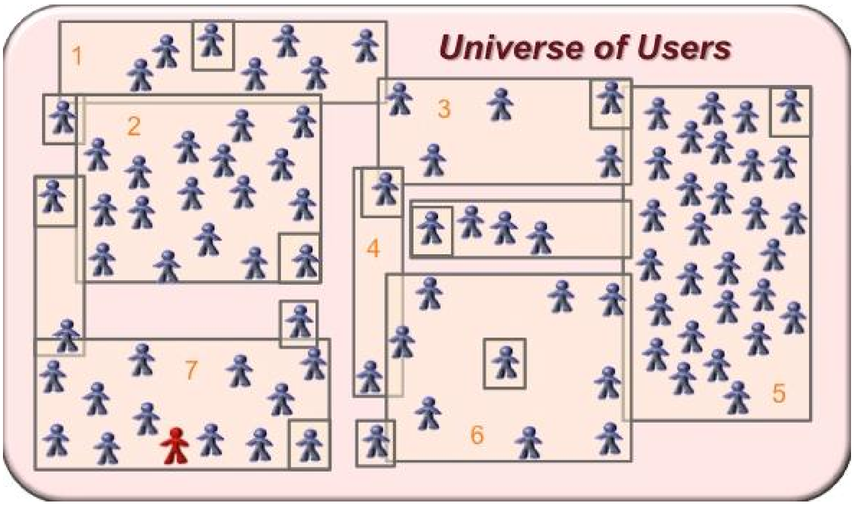
$$\text{Let a group of users (stereotype) be } GU \equiv (UD, p) \mid p \in N \tag{2}$$

The whole set of groups of users for a particular domain will be named universe of users (UU). The UU defined here is in fact the currently available knowledge about the universe of users, stored in the knowledge base of the model's implementation. Since it is a statistical model, any new session will provide new knowledge. Consequently, the description of the UU within any given implementation is dynamic, and the GUs can be updated or new ones can be added. However, an extreme growth of the knowledge base could cause a fatal performance drop. To prevent such drawbacks, some models include upper bounds to the number of information atoms (η) and to the number of groups of users (γ) within a UU. The following expression (3) defines a general UU, taking into account the definition of group of users and the definition of UD:

$$\text{Let a universe of users be } UU \equiv \{GU_i\} \mid i \in N\,[1, \gamma] \tag{3}$$

Even though, the above expression is a general description of a universe of users, this expression might slightly change depending on the user model and its structure. However, those variations are usually based on this general formula. The following image describes this general concept of universe of users giving a perspective of how it looks likes within the structure of a user model.

Figure 35 Universe of Users Distribution



As can be seen in Figure 35, the universe of users of any user model looks like an aggregation of users where each particular user has been attributed to some group of users. Figure 35 represents a general universe of users with just two dimensions. However, it is remarkable that current user models usually deal with multidimensional data. Thus, this figure has to be extrapolated to n dimensions. In addition, each individual group of users might have a different number of dimensions than any other group. Several implementations can be found across the literature but user models try to attribute new users to the most similar group of users in its universe of users. It is also noticeable that some users belong to more than one group of users and that the group of users are mostly overlapped. In fact, due to the usually huge amount of dimension to define a user and a group of users, most of them are overlapped. As can be seen in Figure 35, the current user has been painted in red. When a new user comes, it has to be fitted with the most similar group. In fact there are several possibilities when inserting a new user in terms of the structure of the group. In Figure 35 different groups of users have been numbered. In this case the new user has been inserted in the group number seven. This is a well-formed group, with relatively low overlapping. It might be the case that the current user belongs to group four. This is a mostly overlapped group with just a few users (probably outliers). The third scenario is that the current user falls into group number five. This is a well-formed group though an enormous one. This type of group suffers the problem of certainty degradation due to the amount of users that belong to the group. Most user models have to deal with certainty degradation, however these types of groups are prone to provide poor quality predictions due to the huge degradation. Thus, a universe of users mainly with groups like number two or number seven will provide accurate predictions. However, if the structure of the universe of users is degraded with enormous groups like number five, mostly overlapped groups like number three or four, being the latest one a group of outliers, the quality of predictions will be dramatically affected. Therefore, the structure of the universe of users is crucial for the quality of predictions provided. Poor structured universe of users usually provide inaccurate predictions.

During any interaction, the interlocutor is characterized incrementally as information atoms about him/her are acquired. The atoms are gathered in the current user description state (CUDS), which is formalized as a general UD (1) according to the expression (4).

Let the CUDS be a UD, and its resultant stereotype GU(CUDS) = (CUDS, 1) (4)

When finishing a session, the CUDS is stored as a new group of users (with population 1). Storing every new CUDS in the knowledge base may lead to exceed the threshold (γ). In such cases, the knowledge has to be summarized to meet γ again. Specifically, the most similar pair of GUs has to be merged into a single GU, thus decreasing the size of the UU in a unit. The required procedures to do this are: 'finding the proper pair' and 'merging groups', which are based on the match and the fusion functions, respectively.

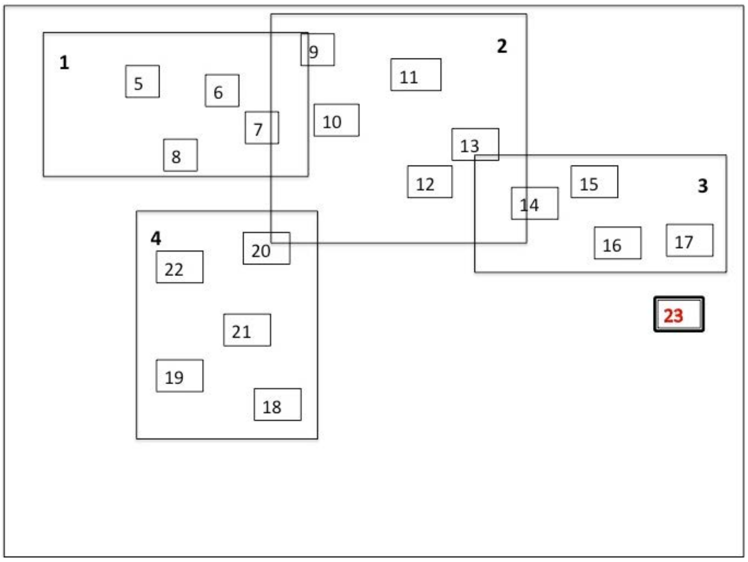
4.1 General functions and concepts

Together with the common definitions detailed above, there exist some general functions that every user model needs to employ. Even though, several different implementations of these functions can be found across different user models, each model relies on these functions to provide predictions. These functions are usually called the match and the fusion function.

4.1.1 The match function

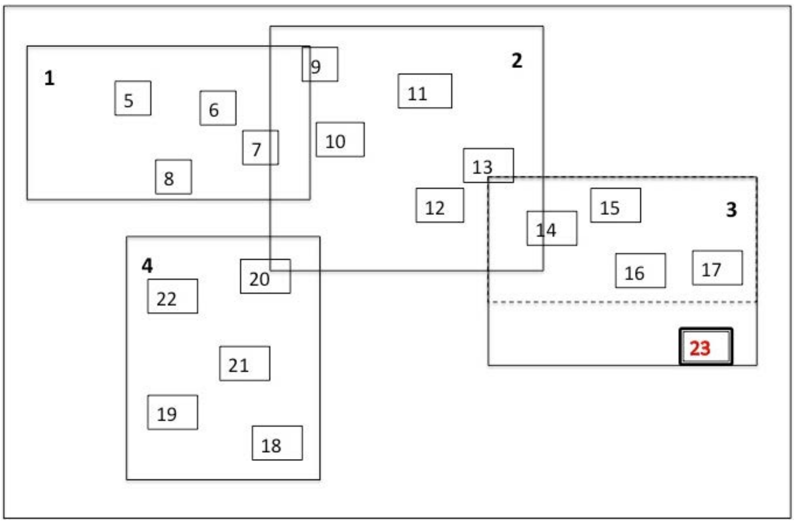
As previously explained and as it can be seen in Figure 35, every new CUDS has to be stored in the knowledge base. Every user model tries to figure out the most similar group of users (GU) in its knowledge base and attach the new CUDS to that group of users. In terms of geometry, the most similar group can be seen as the group that requires least modification to include the new user. Figure 36 depicts the scenario where a given knowledge base has to include a new user, (user 23). Currently, there are four different groups in the knowledge base where the new user might fit. The matching function is then responsible for figuring out which group requires least modification to include the new user. This geometrical approach to different dimensions can be understood as the most similar group. Therefore the final purpose of the matching function is to find the most similar group taking into account the multiple dimensions of the data that usually populates user models.

Figure 36 Match Function



Even though, the case represented in Figure 36 is pretty clear and the most similar group to user 23 is the group 3, the match function must iterate and try to fit the new user with each particular group of users in the database. For the scenario of Figure 36, the resultant knowledge base is detailed in Figure 37.

Figure 37 Match Function Final State

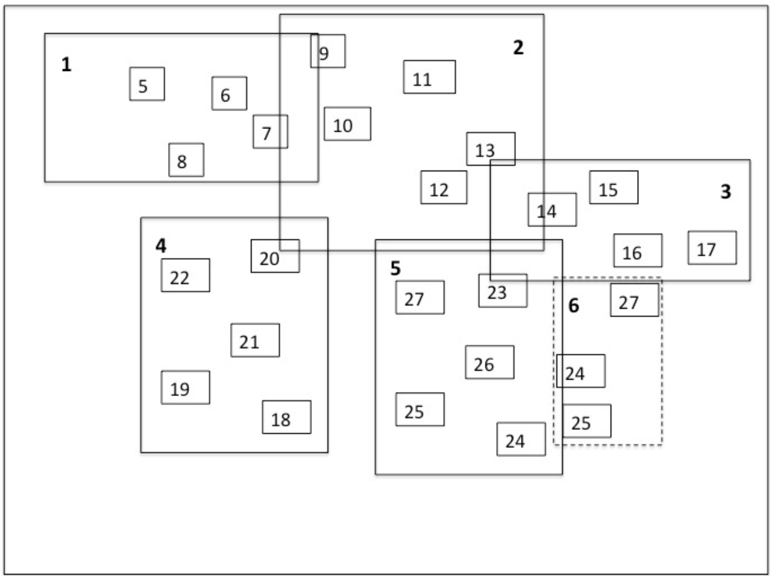


As Figure 37 shows, definitively the group three is the most similar and therefore requires least modifications in the knowledge base to include the new user. Thus, the matching function is responsible for finding the most similar group of users given a particular user description. Therefore, this function provides the most similar group of users given any other user description. That is given a current user description (CUDS), the matching function will provide the most similar user description (UD) within the universe of users of the model. Though, many different implementations of this function can be found, most of them rely on different variations of the k nearest neighbour algorithm. In fact, the problem of finding similarities is a classification problem and any classification algorithm might be used to develop the matching function.

4.1.2 The fusion function

Though, the matching function is in charge of finding the most similar group of users given a user description, the fusion function is responsible for merging two or more groups of users into a new one. This function is usually related with the size of the knowledge base and it is usually triggered when the size of the knowledge base exceeds some particular bound. However, this function is also highly related with the performance of the model. Merging means keeping the size of the knowledge base within some margins and therefore preserving the performance of the model. As it has been previously stated the performance of most user models drops dramatically when the size of their knowledge base grows. It has been documented as the scalability problem and most user models suffer this problem. Figure 38 depicts a case where some knowledge base bound has been set up to five groups and a new group has to be merged. In that case, the fusion function is triggered in order to merge those groups. It is important to emphasize that the fusion function first need the match function to figure out what groups are being merged. Finally, the fusion function is responsible to update the structure of the knowledge base and merge those groups causing the less impact to the knowledge base.

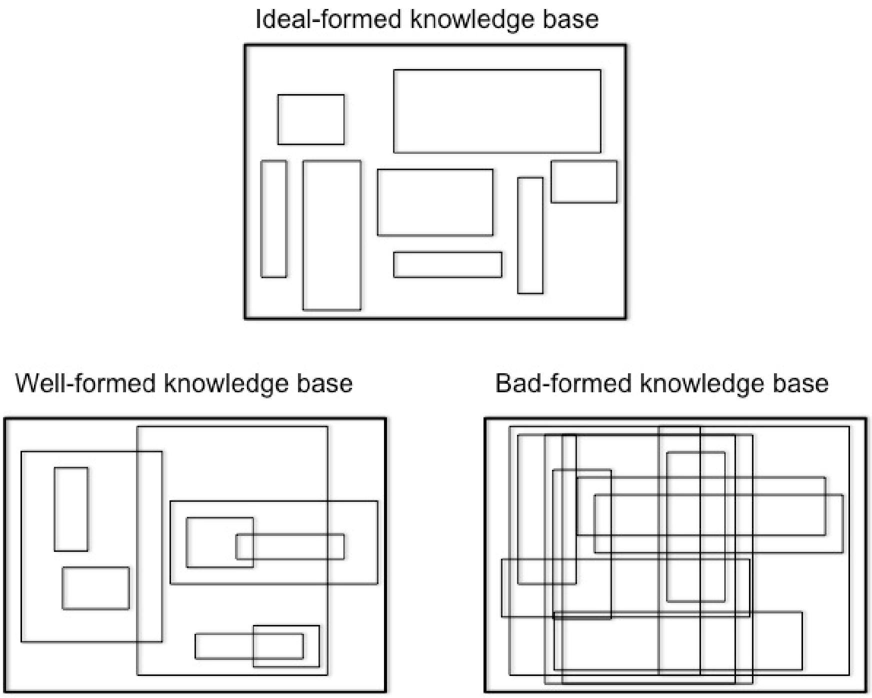
Figure 38 Fusion Function



In Figure 38, once the fusion function is triggered, both group five and six are merged due to their similarity.

Once two main functions have been explained, it is understandable that both functions are highly related to the structure of the knowledge base. That is, every user model relies upon these functions to keep their knowledge base. The knowledge base has to be seen as a component that is in a continuous state flux. Thus, keeping the knowledge base well-formed is crucial for the proper performance of the model. Figure 39 shows two examples of how the knowledge base might evolve to well-formed or bad-formed bases.

Figure 39 Ideal, Well and Bad formed knowledge bases



As can be seen in Figure 39, the top knowledge base is the ideal scenario where there is no overlapping, all the groups are well formed and the amount of empty space is relatively low. Secondly, the left base is a pretty well formed knowledge base where: the overlapping is still low, most of the groups are well structured and no much empty space is missing. On the other hand, the right knowledge base has a bad structure. It is plenty of overlapping and all the groups are affected by certainty degradation. Identifying groups in the two ideal and well-formed knowledge bases is pretty simple; however, the structure of each group is unclear in the bad-formed base. Moreover, spotting the centroids in the bad-formed base is difficult since most of the centroids fall in a small region. In contrast, the well-formed and the ideal knowledge base have clear centroids for each of their groups. It is remarkable that Figure 39 represent the knowledge bases in just two dimension. Current user models have to deal with n-dimensional data. The most dimensions the worst are the structures of the knowledge bases. In fact some metrics to measure how good the structure of a knowledge base is can be computed. First, metric measures the amount of overlapping. As overlapping increases the quality of the recommendations decrease. In addition, high-overlapped knowledge bases are also hard to fit new users. Second, metric computes the amount of empty space this metric is therefore related to the level of overlapping. Even though, the best scenario to perform predictions is described by the ideal knowledge base. In practice, this scenario is unreachable. As previously explained, these models have to deal with multidimensional data and usually with poor formed structures.

4.1.3 Common features

After describing main common concepts and functions of every user model, there are some features regarding the behaviour of a user model that must be described. Before start building any particular user model, some decisions must be taken. It is important to remark that these decisions will affect the performance of the model and how suitable is the model to perform in different scenarios. The most important decision to take refers to how the model performs predictions. For that purpose, three main approaches coexist. First one is the content based approach. Second one is the collaborative approach and the third one is a hybrid approach mixing main features of former approaches. As previously stated, both approaches have great advantages and several disadvantages. Thus, the reasons to use one approach or the other depend on the model and their use. However, according to the bibliography, hybrid user models that combine both approaches usually perform better than other models constructed following one specific approach. These hybrid models used to present the advantages of both approaches and used to compensate the disadvantages of every approach by combining both techniques. The second decision involves deciding how the model compute similarities (match function), two main approaches exist: the Memory based and the Model based. Both approaches differ in the way they look for similar users in the knowledge base. Memory based approach used to employ the whole knowledge base to generate predictions. Thus, memory-based model relies exclusively on similarity measures to match similar people. On the other hand, model based algorithms take a probabilistic approach and build a model that represent users in the knowledge base. Therefore, model based systems employ Machine Learning algorithms to construct the model and rely on these models to generate predictions. Though, these models can be very simple or very complex, model based techniques usually perform better than model based.

Finally, a user model can be context aware or might ignore context. Since context is usually a source of knowledge that can provide better insight to generate more reliable recommendations, models aware of context usually outperform models ignoring context. In case a user model manages context, three main approaches are usually adopted for any user model: contextual pre-filtering, contextual post-filtering and contextual modelling. Since these approaches have previously been explained, in this section they are just mentioned in order to understand that this decision must be taken in advance.

Before going into the details of how each model has been constructed, a mathematical formalization of specific functions developed for this model is given. Though, the mathematical formulation is not the main goal of this thesis, the formulas in each model will be validated and tested. It is also remarkable that most of this formulation is based on well-known algorithms such as: k-means or k-neighbours. Since the formulation employed in every model is equivalent for each common function such as: the matching function, the fusion procedure, etc, comparisons in terms of efficiency can be done. Once main common features and components of a user model have been described, some particular issues concerning each model together with a formal mathematical description of the models is detailed. Hereafter, the user model subjected to knowledge loss will be referred as KLUM while the new approach proposed on this research thesis will be referred as RTUM.

4.2 First approach, KLUM user model

As previously stated a baseline model is needed to be able to measure the improvements of the new approach. This first model follows main principles of traditional user modelling. In traditional user modelling, the knowledge base size is usually limited in order to control its growth. A bounded knowledge base is the reason that causes knowledge loss in this type of models. As the size of the knowledge base grows and reaches its maximum size, some knowledge is probably merged to maintain the amount of knowledge within the limits. Even though, most of these models do not have an explicit knowledge loss, their merge procedure entails an implicit knowledge loss. This is the case of the baseline user model developed for this proposal. Though, there are many reasons to limit the size of the knowledge base, the most important reason is keeping the efficiency of the model within some margins. For that purpose, two bounds used to be set up. The first one limits the total number of information atoms in the knowledge base; while the second one is an upper bound for the number of descriptions of user groups. The first bound is set up to reduce the number of candidates when matching groups; while, the second one lowers the average execution time of each matching process. Though, obtaining the most adequate bounds is not a simple task, in that case these bounds were obtained through preliminary experimentation by observing both the overall performance of the model for different configurations and the individual response times for every interaction. This preliminary experimentation was done using real hardware resources similar to the resources that will be used to evaluate the models.

This baseline model relies upon a few notions (some of them widely spread), which are now to be defined. Therefore, departing from the previous formulas and concepts described and supported by every user model, this particular model relies on some specific concepts and formulas. That mathematical formulation is going to be described.

The internal matching function (M) that the model employs to find the most similar pair of GUs (each pair of GUs in the knowledge base), provides a value within the range [0, 1] measuring the similarity among each pair. The pair maximizing that value will be chosen for merging. The M function proposed here is described in (5).

Let the match be $M: UU \times UU \rightarrow [0,1]$ $M(G_X, G_Y) = \left(\overline{\{ \overline{\mu(x, G_Y)}, \overline{\mu(y, G_X)} \}} + 1 \right) / 2$ (5)

$\forall x \in X \ \forall y \in Y$, where $G_X = (X, p_x)$ and $G_Y = (Y, p_y)$ are groups of users

The M function is defined upon the notion of partial matching (μ), providing a portion of information about the matching. It is used for comparing an information atom (from a GU) with another GU. The result can be positive or negative, that is information about matching in case of positive partial matching or information about knock out in case of negative partial matching. The definition of this function observes two main cases: the disjoint partial matching (special case), and the coincident partial matching (general rule).

In first place, the disjoint partial matching occurs when comparing an atom on a feature that is missing in the other GU (there is no atom defined for that feature). These cases provide no information about the matching, and therefore the result of disjoint partial matching is zero. The rest of the cases are considered as coincident partial matchings, and its calculation has to observe every atom in the GU defined on the coincident feature. Thus, its definition will be based on another function, the single atom match (δ) providing a measure of the match for two individual atoms. The partial match function (6) is defined in Table 6 observing the fore-described cases.

Table 6 Partial match function

μ-match type	Condition	Formula	
Disjoint	given $x=(c, v, s, z)$ and $GY=(Y, pY)$ $\forall\ y=(c', v', s', z') \in Y$, meets $c \neq c'$	$\mu(x, GY) = 0$	(6)
Coincident	given $x=(c, v, s, z)$ and $GY=(Y, pY)$ $\exists\ y=(c, v', s', z') \in Y$	$\mu(x, \overline{GY}) = \delta(x,y)\ \overline{((x,y))}$, $\forall\ y \in Y$	

The definition of the proposed δ function (atoms match) is divided into three different components. The first component is the sign of the matching (7) that is a notion of lace or a knock out notion and is calculated as follows:

$$S = S_x \cdot S_y \tag{7}$$

Being S_x, S_y the signs of the atoms to compare. Even though S defines the sign of the result; the sign of match through two cases should also be observed: when both atoms have the same value (v), the match is equivalent and the sign is positive; on the contrary, if the values are different the match is non-equivalent and the sign is negative. The second component of partial matching function involves weighting the certainties of both atoms. The measure used to weight these certainties is the arithmetic average of both certainties. This component is calculated in the following expression (8), according to the next formula:

$$\overline{Z} = (z_x+z_y) / 2 \tag{8}$$

The third component of partial matching is going to be applied when the equivalency is about the antithesis (both atom signs are negative) or the non-equivalency involves a negative value (any sign is negative). In these cases, the result must be corrected by adding a domain-dependent correction factor (9). The information provided by the partial matching excludes one element of our universe; therefore the correction factor is calculated as follows:

$$f = 1 / (|dom(C_x)| - 1) \tag{9}$$

Once the three components have been explained, its product defines the magnitude of the atom matching formula (10). Taking also into account the cases of coincidence (equivalent and non-equivalent), the function can be defined as shown in Table 7.

Table 7 Atom matching cases

Atom match type	Sign (x)	Sign (y)	Formula
Equivalent: x=(c, v, s, z) and y=(c, v, s', z')	+	+	$\delta(x, y) = \overline{S} \cdot Z$
	+	-	
	-	+	
	-	-	$\delta(x, y) = \overline{S} \cdot Z \cdot f$
Non-equivalent: x=(c, v, s, z) and y=(c, v', s', z') and $v \neq v'$	+	+	$\delta(x, y) = -\overline{S} \cdot Z^{\overline{Z}}$
	+	-	$\delta(x, y) = -\overline{S} \cdot Z \cdot f$
	-	+	
	-	-	

(10)

As stated before, when the number of GUs within the UU exceeds the upper bound (γ) the most similar pair of groups are chosen by calculating the match of every candidate pair. Once chosen, the attention is focused onto the fusion function (11), which can merge two or more user groups (in general, any non-empty part of a UU) into a unique user group. The resulting group will be added while the operated groups will be removed. Thus, decreasing the current number of groups in the knowledge base.

Let the fusion function be $\varphi: P(UU) - \phi \rightarrow GU \mid$

$\varphi(G_1, \ldots, G_n) = GR = (R, p_R)$, where $G_i = (X_i, p_i)$, $p_R = \sum p_i$

and R is $\{(c, v, sgn(z), |z|) \,/\, \forall\, c, v$ meeting $(c, v, \_, \_) \in \iota$

and $z = (\sum_j s_j \cdot z_j \cdot p_i)/p_R \;\; \forall\, i, j$ with $(c, v, s_j, z_j) \in X_i \}$

(11)

It should be pointed out that certainty may drop severely when running reiterative fusions over several groups, producing unworthy tuples with virtually no effect on predictions. Besides, the excessive growth of tuples in the base may produce performance drop during inferences, and because of this an upper bound is set on the number of tuples (η). The purge procedure will simply remove as many tuples as needed to meet that bound again, starting always with the lower certainty tuples.

On the other hand, in order to provide prediction services, the user model is also able to infer values for unknown features of the user. It will do this by matching the CUDS (4) with each group stored in the knowledge base. Then, a value is chosen by applying the selection function (ι). In the following chart (12 and 13), two selection functions $\iota 1$ and $\iota 2$ are proposed. Both of them will be observed in the evaluation section. The first approach is based on the idea that the most similar group will provide most reliable inferences. This approach is called maximum fit inference (12) because inferences are accomplished over the most similar group. On the other hand, the second approach is based on the idea that sometimes the most similar group does not provide the most reliable inference while a slightly different group might provide better inferences (13). This mechanism takes into account the match value and the certainty of the feature. Both proposals will be observed in the evaluation section.

$$\iota_1(c, CUDS, UU) = v \mid \exists\, (c, v, s, z) \in X_i, GU = (X_i, p_i) \in UU, \forall\, (c',v',s',z') \in X_i \rightarrow z' \leq z, \forall\, GU' \in UU \rightarrow M(CUDS,GU') \leq M(CUDS,GU) \quad (12)$$

$$\iota_2(c, CUDS, UU) = v \mid \exists\, (c, v, s, z) \in X_i, GU = (X_i, p_i) \in UU, \forall\, (c',v',s',z') \in X_i, \forall\, GU' \in UU \rightarrow z' \cdot M(CUDS,GU') \leq z \cdot M(CUDS,GU) \quad (13)$$

Such a user model subject to knowledge loss is formalized in KLUM (F, γ, η, M, φ, ι), where F is a set of pairs (feature c, domain(c)) within a wide range of attributes and behaviours characterizing the user; γ and η are upper bounds to the number of GUs and to the number of tuples in the base, respectively; and M, φ and ι are the match, fusion and inference functions, respectively. Notice that, although the set of features and domains F is included for definition completeness, the model can learn new features and domains as they are acquired through different sessions during the system's lifetime.

4.3 Second approach, RTUM user model

The abbreviation RTUM user model stands for R-Tree Guttman, (1984) User Model. Therefore, this model has been built under the principles of the R-Tree structure. In general all functions like partition, fusion, etc are based on the basics of the R-Tree. R-Tree structure has proven capabilities when dealing with multi-dimensional data. In addition, using an R-tree structure over multi-dimensional data prevents employing classical indexing structures that usually provide poor performance when dealing with this type of data. Basically, an R-Tree is a height-balanced tree where the nodes contain pointers to data objects. One of the most important advantages of the R-Tree is that searching only requires visiting a few nodes.

It has been stated that traditional user modelling usually imposes a limit to the size of the knowledge base that limit helps to keep performance of user model within reasonable margins. This is usually called the scalability problem in user modelling. RTUM thanks to the R-Tree structure does not enforces a limit to the knowledge base. According to R-Tree performance (Section 2.2.1.3) this data structure is designed to keep performance even when dealing with large datasets. User models requires tight response times when performing online processes. Such response times has been set to be lower than one second in order to be considered real time performance. User models are required to provide real time performance for those processes that need user interaction.

Even though, RTUM does not impose a limit to the amount of information atom in its database, it limits the number of descriptions (user groups) in each sub-universe of users. It means that the model sub-divides the overall universe of users into several-overlapped sub-universe of users. This refinement occurs due to its structure. R-Trees or rectangle trees divide the space in several overlapped rectangles. The same idea can be extended to the user model problem. According to the definition of universe of users given (3), that space is exactly the same that the universe of users in a given user model. Thus, the R-Tree imposes dividing the universe of users in overlapped rectangles having the minimum intra-rectangle distances and the maximum inter-rectangle distances. Formally a sub-universe of users (14) is defined as follows:

Let a sub-universe of users j of the level i be $UU_{ij} \equiv \{GU_k\} \mid k \in N$ [1,n (14)

According to the design of the R-Tree, new elements (individual descriptions of user sessions) are added to the leaves of the tree. By the way, there are two different types of sub-universe of users. Those in the deepest level of the tree, having individual user descriptions, and those non-leaf nodes keeping user's descriptions representing the set of users in its sub-tree (UUi,j). It is important to emphasize that in this particular model the universe of users is represented with different detail, depending on the level of the tree. Thus, sub-universe of users formed by the root node contains the most general knowledge and that knowledge gains specificity all the way to the leaf nodes. In fact, the conjunction of all the leaves of the tree forms the universe of users for this model. It is remarkable that the user model keeps in the leaves of the tree the overall knowledge gained during the different sessions. The structure enables providing inferences with different granularity depending on the level of the tree. Formally the universe of users (15) for this model is re-written as follows:

Universe of user UU = $\cup_i UUnj$ being n the tree's depth (15)

According to previous definition, it can be concluded that the rest of the information in the tree (non-leaf nodes) is redundant information. Hereafter, user's descriptions in the leaf nodes are called individual users, while users in non-leaf nodes will be called briefs, since they summarize the content of children nodes. The definitions of these elements (16 and 17, respectively) are supported by a common user description (UD), which is the same already defined for the former model (1). The individual user group (16) found exclusively at the leaves has a UD and population 1, while the general user group (17) has population p and a pointer to a sub-tree representing the sub universe of users UUx+1, and summarized within its UD.

Let GU be a group of users in the leaves level, GU ≡ (UD, 1) (16)

Let GU be a group of users in a non-leaf level, GU ≡ (UD, p, $UU_{x+1,y}$) , $p \in N$ (17)

Consequently, every node in the tree (except for the root) has a summary of its information in its ancestor node, and the summary has a pointer associated to that node. Formally:

$\forall\ UU_{x,j}$ with $x > 1\ \exists$ a $GU_{x-1,j}$ in its ancestor node that is the brief of $UU_{x,j}$ (18)

According to (18), $GU_{x-1,j}$ is the group resultant of merging the whole set of user's groups in the node UUx,j.

4.3.1 Merging and Partition

The merging procedure is supported by the fusion function φ (11), which is the same already used in the former KLUM model. While KLUM employs the fusion function to preserve the size of its knowledge base, RTUM uses this function to control the size of its sub-universe of users. Thus, in this case instead of restricting the size of the knowledge base, the upper bounds (γ and η) restrict the number of group descriptions and information atoms (respectively) of every sub-universe of users, that is any node of the tree. However, it is remarkable that while imposing a limit to the size of the sub-universe of users the model does not imposes any limit to the whole knowledge base size and therefore keeps all the information stored in the knowledge base. The upper bound η limits the number of information atoms that any sub-universe of users (tree node) can contain. However, this definition is not fully consequent with the idea of avoiding knowledge loss. It might be the case that a leaf node of the tree structure would exceed the upper bound η. In that case, if some information atoms were removed, the model will fall into knowledge loss. Note that it would not happen with any other node since the information in the non-leaf nodes is redundant information. Therefore, RTUM does not enforce this limitation over the leaf nodes. Leaf nodes trend to be smaller than any other node in the R-Tree structure and consequently skipping this limitation not only does not point to a huge growth of the knowledge base, but also enables being consequently with the idea keeping all the information in the knowledge base.

Even though, RTUM has several differences with traditional user models (KLUM) in the form that it treats upper bounds, main difference is that RTUM not only summarizes knowledge, but also distributes knowledge; and after that, summarizes knowlege. That is, when any node grows and exceeds the upper bound (γ), the knowledge in this node has to be distributed along the R-Tree structure and therefore summarized. Exceeding the γ limit means that the number of groups in that particular node is over than allowed. In that case, the node is to be partitioned in two (or more) nodes. Thus, the former node is replaced with new nodes. The elements of such node are therefore distributed between those new nodes. Finally, the summary of the old node, located in the parent node, will be replaced with the summaries of the new nodes. Each new summary has attached a pointer to the new node.

4.3.2 Insertion

On the general idea of how the model works and how it manages to keep generated knowledge. A review of the overall insertion procedure is being detailed. As long as the model is built under the principles of the R-Tree, the insertion process will be done according to that basis. Inserting in an R-Tree is pretty much similar than inserting in a B-Tree, new records are always inserted in the leaf nodes. Therefore, leaf nodes may cause an overflow, such an overflow leads to splitting the node (distributing) and after that propagating that split up along the tree (summarizing). However, before inserting the node and before splitting procedure is triggered, it is needed to find the proper node to perform the insertion. Thus, the insertion algorithm starts exploring the root node to find the most similar record to the new record (matching function 5). Emphasize that the formulation for the match function (and its subsequent supporting functions) is the same that the one applied to the previously explained model subject to knowledge loss (KLUM).

After that, the algorithm goes one level deeper for the proper pointer and repeats the same search, this process is repeated until the leaves level is reached. Once one particular node on the leaves level is reached, the new record CUDS (4) is inserted in that node. Notice that the initial population of that particular CUDS will be one. If the node has room for a new record, then it is inserted and the changes are propagated over the tree. On the other hand, if the node does not have room enough for a new entry, then the split procedure is triggered. The following algorithm states how the insertion occurs in any generic R-Tree.

Step	Description
1.- Insert	Find position for new entry E; invoke ChooseLeaf to find a leaf L to place E.
2.-Add record to leaf	If L has room for a new entry then add E. Else invoke SplitNode to obtain L and L' containing E and all entries in L.
3.-Propagate changes	Invoke adjust tree on L and on L' if a split was required.
4.- Grow tree	If split propagation causes the root to split, then create a new root whose children are the two resulting nodes.

As can be seen, the algorithm match up the previous explanation; however, one thing is missing on the explanation. As previously explained, any insertion might cause one or more partitions. In general, one partition produces new nodes replacing the former node in the same level of the tree. There is an exception to this case, if the node that is being partitioned is the root node, the new nodes are created in a new level making the tree grow one level.

Thus, as previously stated, this model not only summarizes; but also distributes content along the tree structure, for that purpose the model relies on a distribution function ρ (19), which provides a distribution of the elements within a sub universe of users into several ones (at least two) by following some criteria (specific of each implementation).

$$\rho: UU_{x,y} \rightarrow \{UU_{x,z}\} \mid z > 1, \bigcup UU_{x,z} \equiv UU_{x,y} \text{ and } \forall\, i,j\ UU_{x,i} \cap UU_{x,j} \equiv \varnothing \qquad (19)$$

According to (19) and the definition of R-Tree, the structure only gains new levels (depth growth) when the root node overflows. Even though some overflows in the root node can happen, they might turn to happen very frequently if the rest of the nodes in the tree are mostly empty. To avoid a quick depth growth of the tree that might lead to performance drop, the nodes of the tree must meet a minimum occupation requirement (k_{min}). Given that an overflowed node has γ+1 elements, the maximum cardinality of a sub universe resultant from partition will be $\gamma+1-k_{min}$. In fact, this is a requisite applied by the specific implementation of ρ. In the following formulation (20), a proposal of such implementation is provided in order to illustrate the general definition of ρ.

$$\text{Let } \rho_0(UU_{x,y},\varnothing,\varnothing) \equiv (UU_{x,y} - \{GU_a, GU_b\}, \{GU_a\}, \{GU_b\}) \mid \quad (20)$$

$$GU_a, GU_b \in UU_{x,y} \wedge \forall GU_c, GU_d \in UU_{x,y} \text{ is } M(GU_a, GU_b) \le M(GU_c, GU_d)$$

$$\forall i \neq 0, \text{ let } \rho_i(A,B,C) \equiv (A-\{GU_a\}, B', C') \mid$$

$$GU_a \in A \wedge \forall GU_b \in A \text{ is } \max(M(GU_b,B), M(GU_b,C)) \le \max(M(GU_a,B), M(GU_a,C))$$

$$\text{with } (B' \equiv B \cup \{GU_a\} \wedge C' \equiv C) \text{ iff } |C| \ge \gamma+1-k_{min} \vee (|B| < \gamma+1-k_{min} \wedge M(GU_a,B) \ge M(GU_a,C))$$

$$\text{and } (B' \equiv B \wedge C' \equiv C \cup \{GU_a\}) \text{ in other cases}$$

$$\rho(UU_{x,y}) \equiv (UU_{x,1}, UU_{x,2}) \leftarrow \rho_n(\rho_{n-1}(\dots \rho_0(UU_{x,y},\varnothing,\varnothing)\dots)) \equiv (\varnothing, UU_{x,1}, UU_{x,2}),, \; n = |UU_{x,y}|-2$$

After creating these two (or more) new nodes, their briefs are obtained as the fusion of all its content (all the elements within). These briefs with the correspondent pointers to the new nodes are inserted into the parent node. Finally, the overflowed node (and, consequently, its brief at the ancestor) is removed. This general rule can be applied to every node except from the root node. If the root node overflows, it will be partitioned just in the same way. But in such case a new root node must be created, containing the briefs and pointers to the new nodes.

Notice that by increasing the number of group descriptions in the ancestor, it may be overflowed in turn leading to a new partition process, and so on till the root. Even though multiple partitions would affect the system's performance, this is an infrequent scenario in R-trees. In fact, even regular partitions are infrequent due to the R-tree structure. However, those insertions triggering a partition process assure that the tree structure is updated or at least that the branch affected by the insertion is updated. Notice that those insertions that do not trigger any partition may lead to obsolescence of this branch of the tree. As has been previously mentioned, every node has a summary in the parent node, if a new insertion does not cause a partition, the summary of the node where the insertion happens won´t be updated. In this scenario, the parent node has a summary that does not take into account one of the record in the child node. Consequently, its current parent node won't be fully updated and so on until the root node. That is, one branch of the tree is not totally updated because an insertion does not cause an overflow. One simple solution to this problem that might lead to obsolescence of the tree is forcing the update of the parent node and successively of every single node until the root. Though, this is the most straightforward solution, it may require high computational cost to perform all the updates. However, the number of updates can be reduced controlling the number of insertions and forcing the updates every "x" insertions. Anyway, it should be pointed out that the eventual loss of performance due to inserts does not affect any critical process, since insertion is an offline process (it runs once the user is disconnected).

4.3.3 Inference

The final purpose of the user model is providing reliable predictions or inferences. However, reliability and performance are not truly compatible. Since inference is an online process that involves the user, efficiency and precision are very important. Processing several nodes of the tree to provide an inference is usually unaffordable even when just taking into consideration one single branch of the tree. Thus, the inference process must focus on a unique node to provide efficient responses. For that purpose, the model incorporates the idea of current window. The current window is basically a pointer to a tree node that is being added to the CUDS (21). At the beginning of any inference, that pointer is set up to point the root node. Thus, any inference starts always in the root node and might descend through the tree depending on some circumstances. The CUDS is matched with the different GUs in the root node, the function (τ) is used to figure out the best match. In fact, (τ) is responsible for determining when a match is good enough and therefore trustworthy. Once again the definition of that function may, and probably, vary from one model to another. It may be that simple as a certainty threshold, or different thresholds for each level of the tree, a threshold over the difference between the two best matches and so on. Anyhow, once the match satisfies the function the pointer of the current window must be updated with the node summarized by the best match. That is, the current window descends one or more levels for the best-matching branch. New matches will be performed in that node and so on until that ideally the leaves level is reached.

The CUDS is a UD, its stereotype and the current window GU(CUDS) = (CUDS, 1, UU_{ij}) being i the root level at the beginning of every interaction. (21)

Once the sub space of users is selected, and having the matches already calculated, the inference is accomplished by applying a selection function (12, 13), as in the former KLUM model.

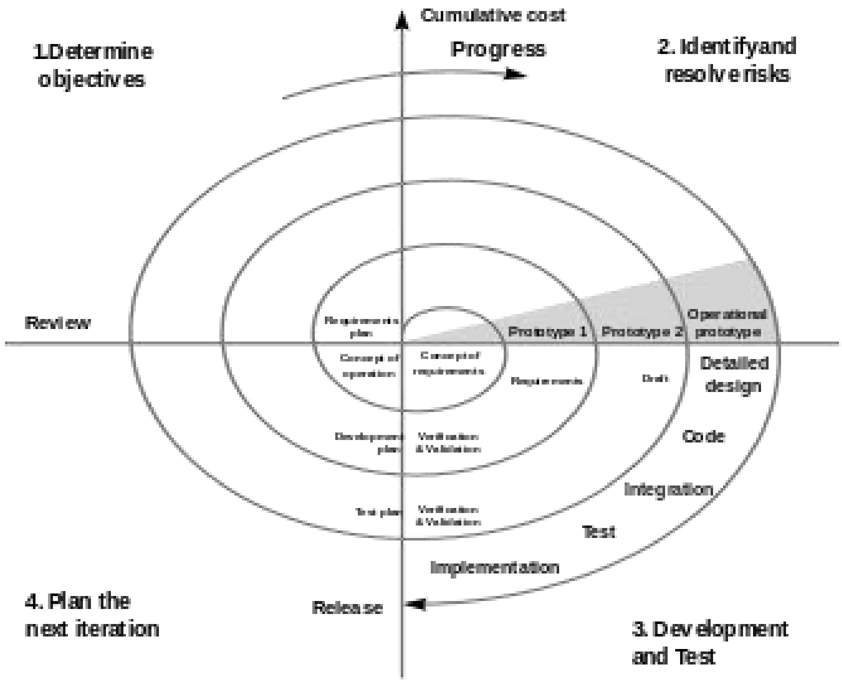
4.3.4 Summary

In summary, RTUM user model is based on an R-Tree structure and is able to avoid knowledge loss. Formally, the model is defined by RTUM(F,γ,η,M,φ,ι,ρ,τ), having the same elements as in KLUM (in general as any traditional user model) but adding two new functions. The two new functions are needed because of the R-Tree structure the first one is the distribution function ρ. This function is in charge of partitioning overflowed nodes having in mind that this model not only summarizes knowledge but also distributes knowledge. Secondly, the match test τ is responsible for updating the current window during the inferences. Although the model is defined by generality, examples of general functions are added to illustrate the proposal and to depict a specific implementation, which will be evaluated in the following section.

4.4 Software Development Methodology

This section provides an overview of the software development methodology employed during the development of current proposal. There are many different software methodologies that present advantages and disadvantages depending on the scenario. For this particular case a spiral methodology has been chosen due to its flexibility and capabilities to refine and improve the prototypes. This methodology is based on several iterations. Activities and tasks may vary from one iteration to another (flexibility). On the other hand, each new cycle is usually set up depending on the results of former iterations. Therefore the methodology provides a good framework to develop a base prototype and improve this prototype through several iterations. In addition, new tasks and activities can be included on each cycle depending on demands and previous results. Figure 40 depicts a general spiral methodology containing some common task for each iteration.

Figure 40 Spiral Methodology



As can be seen, the idea behind this methodology is a continuous improvement through different iterations. Though, the methodology proposes different cycles, the number of cycles for each software development has to be decided. In the case of current proposal five different cycles are proposed in order to complete the whole software development and evaluation.

4.4.1 First approach

As previously mentioned current proposal involves the development of a baseline model to make comparisons between the new approach and this model. Main goal of this cycle involves the research and development of this model. Tasks included on this cycle are the design and analysis required to create the model, a first development of the model and a validation to obtain feedback about the acceptability of the model. The analysis and design phases involves creating the model and making decisions about the structure of the model and its main principles. There are several decisions that will affect the performance of the model. First one involves deciding the structure of the model. Since this model has to meet the principles of classical user modelling, it has a simple structure based on one node. This node is responsible for storing all the sessions and trim knowledge if needed. Regarding the behaviour of the model, it has to be developed to be a hybrid model providing both: content-based and collaborative capabilities. Concerning context the model must be context-aware since context is an additional source of knowledge that might improve the performance of the model. Thus, the analysis and design phases yield a hybrid and context-aware model based on experience and capable to perform in different domains.

After this phase, this cycle involves the development of the first approach to the solution. In terms of technology, the model will be developed using an Oracle table to maintain its one-node structure. In general, Oracle database will be used to develop the model and its main procedures. In addition, Java code will be embedded in the database to provide additional functionalities that cannot be easily implemented in PL/Sql. The decision of taking Oracle 11G as a database management systems comes from the fact that this database provides a framework to embed Java code. Embedded code provides better performance since large amounts of data does not have to be sent over potentially slow networks. Consequently, the majority of the former formulation for KLUM will be developed in Pl/Sql while Java will be used for some particular pieces of code. Thus, for this model the match function and distance functions will be developed in Java. The remaining procedures and the whole structure of the model will be developed in Pl/Sql.

Finally, the validation of the model is needed to guarantee the consistency of the mathematical formulation. Equivalent mathematical formulation will be used in the following models in order to establish comparisons among models. A small set of experiments to check the efficacy of the model will be run.

According to the main tasks and principles of this first stage, second stage must be oriented towards: analysing the results of this cycle; proposing improvements or amends to this model; and starting the creation of the new prototype.

4.4.2 Second approach

As previously mentioned this stage starts analysing the results of former cycle and proposing tweaks in terms of how the models perform. Even though, results obtained are in general satisfactory, a small amend in terms of how the models treats context was proposed. There are different ways of building a context-aware model, pre-filtering, post-filtering and modelling. Most user models employ pre-filtering to be aware of context. That means include context before providing recommendations. The model was tuned to accept pre-filtering and contextual modelling. That is not only pre-filtering but also creating a whole model where knowledge in the database merges with context to potentially provide better recommendations. Once results of previous cycle are analysed and improvements proposed, this second cycle focuses on the analysis and design of the new user model proposal together with its development, validation and evaluation.

Unlike former KLUM, this new approach has to provide new capabilities such as: the ability to deal with scalability problems or knowledge loss. Similarly to previous model, some decisions have to be made in order to outline the model. First decision involves deciding the structure that is going to support the model. This is one of the most crucial decisions since the structure is not only responsible of supporting the model, but also affects how the model behaves and its capabilities. In order to provide the model with advance capabilities an advanced structure is needed. In this case an R-Tree will be used to build the model. R-Trees provides functionalities and capabilities that are required for most user models. R-Trees are able to deal with multidimensional data and provide and advance structure where performance is not affected by the growth of the knowledge base. Unlike KLUM this new approach consists of several nodes forming an R-Tree. Concerning the behaviour of the model, it will follow the same principles as KLUM that means a hybrid model, context-aware providing context pre-filtering and context modelling.

Alike RTUM, this model will be developed combining an Oracle database and embedded Java code. In terms of mathematical formulation, same or equivalent formulation as the one used in former KLUM will be used. Thus, during the development of this model pieces of code developed for KLUM will be re-used.

Similarly to precious cycle, this cycle involves a validation and a small set of experiments in order to guarantee the acceptability of the model and check its efficacy. In this case, these experiments are also intended to check the consistency between the two models.

This cycle also involves building an evaluation framework suitable for the user modelling problem. For that purpose, an evaluation dataset must be provided together with metrics, experiments and an evaluation methodology. Therefore, this task involves: the selection of the datasets, the selection of the metrics and the selection of the experiments that will be in the benchmark.

The final step of this cycle involves evaluating both KLUM and RTUM models with preliminary experiments to decide their best set-up. After deciding a set-up for each particular problem in the evaluation framework, that set-up will be used when facing the model to that experiment. The set of preliminary experiments must be complete and provide feedback to characterize the model to perform in different scenarios.

According to main targets in this cycle, next cycle should: review the results obtained, provide amends, improvements and perform new evaluations.

4.4.3 RTUM refinement

The third cycle of the methodology starts analysing previous results and proposing tweaks to both models and to the evaluation framework. A new refinement of former RTUM should be developed. This amend will provide RTUM a new advantage by getting the most of the data through a little understanding of the domain. That means modify how the model behaves depending on the domain and more concretely depending on which features will come first during an average interaction. First step in this cycle involves the development of this new approach over RTUM model.

Second task after that development involves validating the new approach and its new components. Alike KLUM and RTUM, this new approach will be tested again a small set of experiments in order to prove its consistency and efficacy.

Finally, an evaluation of the model has to be performed in order to measure the gap between previous models and the new approach. Even though, this new approach s slightly different to former RTUM it has to be evaluated under the same circumstances that previous models to obtain reliable data about the differences in performance. At the same time, the three models will be evaluated in another domain to check their performance and their ability to run over different domains.

The fourth cycle of the methodology should take care of reviewing results yielded by latest evaluations, propose new refinements and evaluate those new approaches.

4.4.4 Final improvements

According to previous cycles, this fourth iteration starts analysing the results obtained in the previous cycles and proposing modifications, probably in terms of better understanding of the data. Through this final cycle, it was observed a certain lack of semantic support, often neglected by most approaches of this kind. Indeed, there exist semantic relationships between labels introduced into the model as *user features*. It is not only a matter of dealing with synonymy and multilingual bases, as this and many other systems already implement. It should be dealt going further into the semantic links of different user features. On one hand, many valuations of a feature are performed on a numeric metric, on which there can be calculated *distances* between different valuations. For example, when dealing with users aged 45, it could be useful to apply some knowledge on users aged 44. Distances between values can be obtained even on discrete domains with natural language labels, as long as, there exists a parallel numeric domain to which stablish some equivalence (probably by means of fuzzy logic trapeziums (Zhao and Bose, 2002)). Besides, two different people might be classified as adults but is not the same an adult aged 32 than an adult aged 45 and probably their preferences and likes will be quite different. On the other hand, there exist functional dependencies inside the data, between different concepts, regardless of the people on which they are applied. Thereby, a child lacks of driving license, for example. These dependencies can be exploited to gain knowledge about the data and feed the model with this new knowledge.

Such conclusions lead to propose the support of a strong ontology to the statistical user model as demonstrated by Heckmann, et al, (2005). This ontology can provide distances for features such as ‘age’ and semantic relationship between concepts, empowering the statistical user model performance. Finally, after collecting the information about the domains and providing the model with such knowledge, the model has to be evaluated under the same circumstances than the previous models.

4.4.5 Final analysis

This final cycle does not involve any development but it is necessary to analyse the results provided in previous cycles and to make a comparison of all the results obtained during the whole evaluation process.

5 Evaluation

This section provides an overview of the evaluation process over the three previously described models. The chapter starts with an introduction that explains main targets of the episode and relates the evaluation back to the state of the art. Next the purpose of the evaluation is explained and analysed. Right after, the methodology used to evaluate together with main metrics will be explained. Following, the reader can find the design of the evaluation process including the baseline experiment, the setup of the models and the list of experiments proposed. Finally, the results obtained during the evaluation will be depicted, analysed and discussed, debating main reasons that lead to those results. To conclude the evaluation chapter a complete discussion will be provided.

5.1 Introduction

Though most research thesis have the majority of its workload on the development phase, this proposal has a different distribution being the evaluation the stage with more workload. After developing the models each model has to be evaluated in different scenarios and with different parameterizations. This diversity of models and scenarios allows us to demonstrate the ability of the models to generalize and perform in different domains and in general, measure the improvements of the new approaches.

Main purpose of this chapter is to provide a complete overview about the performance of the models together with a discussion and analysis of the results. However, this chapter tries to go further and at the same time, solve main problems related to statistical user modelling evaluation. Those problems usually derive from the lack of proper mechanisms to complete the evaluation. That means, standard evaluation mechanisms that allow comparing different models and different evaluations. This deficiency causes that most researchers borrowed metrics from other areas e.g. Information Retrieval of Machine Learning while some others use to employ their own metrics. The final outcome is an amalgam of metrics and methodologies that make most evaluations difficult to compare. Furthermore, the area of user modelling not only lacks of metrics for evaluation but also needs proper datasets, experiments and methodologies to make evaluations comparable. Therefore, there is a need of a complete and public evaluation framework to provide researchers and developers proper tools adapted to the problem of user modelling. Those tools to evaluate user models might be taken from other disciplines but they have to be adapted to the particular problem of user modelling. Even though, an evaluation framework is usually designed over some particular domain, that means a specific dataset, the metrics, experiments and methodologies must be domain independent. That means, every metric, experiment and methodology can be exported and applied to any other domain being confident that those tools will work on that domain. Therefore, there is a need of proper tools to evaluate any user model.

However, most user models are specifically designed for some particular domains. It means that those models are likely to provide better performance over that domain than over any other domain. Thus, testing those models only over customized domains does not allow comparison among different models.

On the other hand, the area of user modelling lacks of agreement in terms of evaluation purposes. For instance, some evaluations focus on measuring the quality of recommendations while some other try to measure how often the systems lead to wrong choices. Some works may measure large errors between the predicted and the proper item Shardanand and Andmaes, (1995), other work may suggest that properties different from accuracy might have higher effect on users satisfaction and performance. For that reasons, another important challenge in this area is selecting proper and re-usable metrics that allow researchers to compare evaluations.

To conclude, the evaluation of any user model is hard and compare results with another model in the area can be even harder. The lack of proper mechanisms to evaluate user models has brought researchers to create new metrics for almost every evaluation. This chapter tries to solve main problems related to the evaluation of user modelling at the same time that provides and analyses the results of different evaluations. Those evaluations have been designed to provide reliability at the time of measuring the improvements of the new approach proposed in this thesis compared to previous approaches.

5.2 Evaluation goals

Most predictive statistical User Models seek characterizing the interlocutor for further proposing new values to unknown features, likes or future behaviour. Even though this evaluation proposes several goals that will be detailed along this section, since this work involves the proposal of an enhanced knowledge base preventing knowledge loss, the main goal of this evaluation should be to measure the advantages of the new approach RTUM when comparing with a model subjected to knowledge loss like KLUM. Indeed, every evaluation chapter must tackle other issues that provide completeness to the chapter and to the applied evaluation methodology itself. Among those issues this chapter provides a complete evaluation framework to standardize user models evaluations and then, shows several evaluations performed on different domains and with different parameterizations in order to seek diversity during the evaluation and more perspective in the results analysis.

Specifically the models presented during this proposal will be evaluated on four different domains. Each domain has a different structure and characterization. As previously stated, one of the goals of this proposal is to develop a domain-independent user model. Though, this goal has been addressed during the research and development phases the evaluation must prove the success of the approach. The first domain involved in the evaluation is a real and open domain. It means a real application case domain that has no validation against any ontology or thesaurus. Evaluations over this typology of domains use to provide a good feedback in terms of research but they are usually not used in real systems due to their complexity. It should be noted that systems of this kind are usually supported by a thesaurus or a lexical-base with semantic labels (ontology model) to match concepts and terms and reduce the dependence on the domain. The second domain taking part of the evaluation is a bounded domain that unlike the first domain has been machine-generated. Bounded domain means that some of the terms and concepts within the domain have been validated against and ontology. This validation establishes some naming conventions among terms and concepts that ease the life of any system using that domain. The third domain comes from one of the Spanish research projects that have supported the development of this thesis: Cadooh (TSI-020302-2011-21). This is a real case domain that, unlike the previous domain, counts on ontology model to map terms and concepts. The latest domain has been obtained using the API provided by a well-known social network. Even though this domain is not validated by any ontology model, the social network provides this validation and the domain is considered as a bounded domain. Note that, as previously mentioned bounded domains are prone to provide better results since ontology validation provides standard naming conventions within the domain. Table 8 shows main features of each domain in order to allow a comparison of the complexity of each domain.

Table 8 Domains description

	Open domain	Artificial domain	Benchmark domain	Social domain
Number of users	100.000	100.000	40.000	2.810
Number of features	23	22	29	18
Maximum feature cardinality	177.851	90.196	53	76.746
Minimum feature cardinality	57	7520	2	2
Average feature cardinality	41.788	31.967	7	5.492
Maximum features per user	21	22	27	20
Average features per user	11	21	22	14
Minimum features per user	3	18	20	5
Domain default value	0.01%	0.1%	16%	0.45%
% Multi-valued features	81.8%	24.2%	3.33%	89.8%

According to Table 8 there are significant differences between the open domains (open domain and artificial domain) and the benchmark and social network domain. The maximum cardinalities are higher for those domains as it is the average feature cardinality but the most important features to consider are the domain default values and the percentage of multi-valued features. These two parameters typically have a massive impact on the performance of the models since they increase the complexity of the domains. It can be seen that the domain default values (naïve algorithm) is much higher for those real application domains while the values for open and non-validated domains are significantly lower. On the other hand, the percentage of multi-valued features is also crucial to understand the complexity of each model as multi-valued features are inherently harder to predict than those single-valued features. In this case the social network domain has the biggest percentage thus indicating the complexity of this model.

As aforementioned, one important goal of this evaluation chapter is to address the problems that arise when evaluating a user model. In order to tackle those problems, a complete evaluation framework is been developed and proposed. Among the four previously presented domains, the third one has been selected to create the evaluation framework. This domain comes from a real application case and is also validated against ontology. This fact makes it the most suitable domain to build the evaluation benchmark around it. The main purpose of this benchmark is to provide a framework to evaluate and measure the performance of any user model together with making those evaluations replicable and comparable.

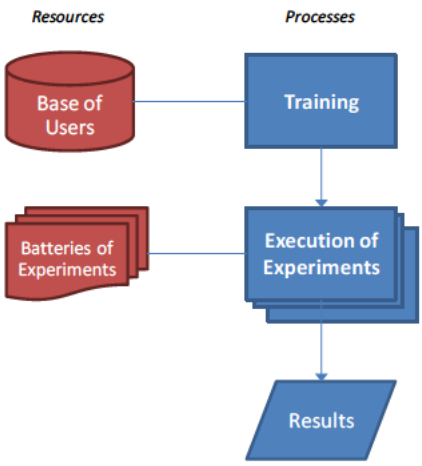
5.3 Evaluation Methodology

Though user modelling is a widely spread multidisciplinary area, the evaluation of user models is still a non-standardized sub-area. Evaluating user models is an inherently difficult task. Moreover, it has become more and more difficult due to lack of standards and evaluation frameworks. The absence of proper tools to evaluate user models has led to non-replicable and non-comparable evaluations. Main reasons behind this amalgam are difficulties to select a dataset to perform evaluation, problems identifying the goals of the evaluation, lack of metrics to decide what has to be measured; and, finally, non-standardized methodologies to run the experiments.

The problem of selecting proper datasets to evaluate refers to the ability of some user models to perform well over some datasets and poorly over different datasets. Similar problems arise when stating the goals of the evaluation. Though several evaluations might have the same goals, the lack of standards to measure those goals make very difficult to compare any pair of evaluations. In addition, the same problem happens when referring to the metrics used in the area of user modelling. Most of those metrics have been borrowed from other disciplines like for instance Machine Learning or Information Retrieval. Though, this might be a good practice, those metrics have to be adapted and standardized to the problem of user modelling. Indeed, the lack of regulation has turned out into an amalgam of coexisting metrics. Furthermore, it can be stated that almost any new evaluation introduces a new metrics or new amends over existing metrics.

Thus, there is a lack of datasets, experiments, metrics and methodologies adapted to the problem of user modelling. In order to fill the gap existing in the area, a complete evaluation framework is proposed, yet observing most common techniques and metrics of similar works, trying to converge in a globally useful evaluation tool. Such evaluation framework is built around a real case domain, from which the knowledge acquisition is validated by means of an ontology model, and which datasets can be published (making it a perfect candidate to build the benchmark). First step when building an evaluation framework is to define the overall evaluation methodology. This methodology has been specifically adapted to the problem of user modelling but it has been designed to be usable in any user-modelling problem. Figure 41 depicts the general evaluation methodology.

Figure 41 Schema of the Evaluation Methodology



According to Figure 41 the evaluation of a user model involves certain resources such as: a base of users (datasets) and a set of experiments. Indeed, those resources should follow some structures that have to be designed and adapted to the problem of user modelling. Furthermore, some processes take part of any user model evaluation and define the data flows shown in the previous figure.

Therefore, the evaluation departs from a base of users and a model. In order to evaluate a user model it is necessary to train the model. Once the model is trained it can be tested against proper batteries of experiments. A portion of users is selected to train the model and some users form the batteries of experiments. Once the model is trained it is tested with the experiments and produces the results of the evaluation. Finally, those results have to be compared and analysed to extract proper conclusions.

5.3.1 Experiments definition

The base of users has to comprise two datasets of samples. First one is involved during the model training and the second one supports testing the model. Even though, two different datasets are used during the evaluation, the test set will be used only for evaluation purposes while the training set is used to train the model and might be used for initial evaluations. Each sample in a dataset is defined as a user description including its full characterization. That is, a set of facts concerning a sole individual (each fact regarding a feature of the user and the value that it takes for this particular user). Notice that any given (multivalued) feature could appear several times (denoting several concurrent values for the same feature and user). Another crucial step when building an evaluation framework is defining the baseline experiment. Once the definition of the baseline experiment has been stated all the experiments will be defined according to this main experiment.

For this benchmark the baseline experiment involves taking a sample that will be named as the current user and selecting one of the facts (features) to be the inference goal. Therefore, once the goal is fixed the evaluation consists of incrementally feeding new knowledge to the model. That is start querying the model about the goal fact. Feed a new item of knowledge to the model and query the model again. It should be remarked that every time the model receives a new portion of knowledge it has more information to refine predictions, and consequently the model should get closer to the goal. Thus, an isolated experiment consists of as many iterations of the pair 'learn fact' + 'infer' as required till the inference is successful (or until there are no more available facts for that user; except for the goal-feature, of course, that will be never fed). Note that feeding the goal fact will lead to biased experiments, as it should easily conduct the model to the goal and move the experiment away from real application experiments.

An alternative experiment could complete always all the iterations, detecting the eventuality of losing the correctness of the inferences when new facts are fed. In addition, a set of experiments is needed to complete the evaluation process as depicted in Figure 41. Any set of experiments must contain a feature stating whether it belongs to the training set or the test set. In addition, other features such as: the number of samples, the statistical significance intervals, and the order to provide new facts to the model or the subset of facts to select the goal fully describe the experiments.

5.3.2 Metrics

The metrics within the evaluation framework provides an idea of what has to be measured in order to provide the results of the experiments. Those metrics are one of the clues to make results of any evaluation replicable and comparable. As previously explained, most techniques used to evaluate statistical user models are borrowed from other areas where non-specific metrics for statistical user models have been defined. In the case of the benchmark, the set of metrics proposed to evaluate are:

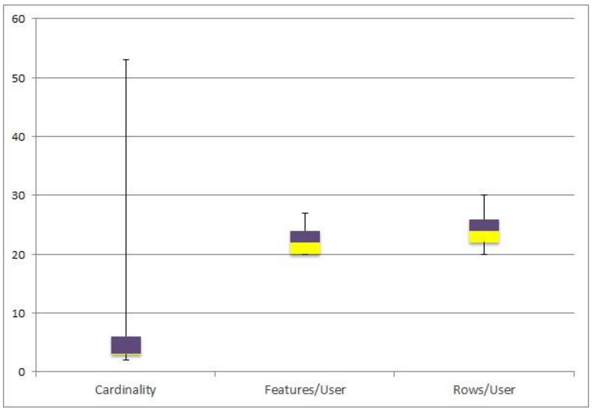
1. Inference success (if the model infers the correct fact).
2. Number of iterations required to get the correct fact.
3. Certainty of successful inferences.
4. Certainty of unsuccessful inferences.
5. Response time.
6. Success rate (whole evaluation).

All these metrics are briefly explained next. The inference success means checking whether the model predicts the correct fact or not. Second metric involves counting the number of iterations required to hit the goal. To preserve the validity of the results through metric (2), the model should not be fed back with the results of its inferences (either success or fail). Thus, at next iteration the model can infer again the same (unsuccessful) value unless the newly fed fact about current user makes it to change its consideration. Nevertheless, another method and metric can be posed by feeding back the model and checking how many iterations requires to success. Metrics (3) and (4) measure the certainty of successful and unsuccessful inferences. User models are set to be real time applications and therefore measuring the response time is important for any user model independently of the application of that user model. The response time shown by the inference process through each iteration is also observed (response time for a single inference, regarding the amount of already-acquired knowledge on current interlocutor). Finally, the success rate is observed as the number of correct predictions divided by the number of users in the test set. In summary, all these metrics come to alleviate the lack of standardization while evaluating and comparing two user models implementations.

5.3.3 Evaluation Framework Description

As previously stated the benchmark has been obtained from a real domain used in the Spanish research project Cadooh (TSI-020302-2011-21), from where over $4 \cdot 10^4$ samples (anonymized user descriptions) were acquired. All features within the benchmark have been discretized in order to go from qualitative values to quantitative values and therefore smooth the impact of those features in any user model. The set of samples was split into a training set (28.000 randomly chosen samples) to train the statistical user model before its evaluation, and a test set (the remaining 12.000 samples) containing enough samples to support different confidence and interval levels that assure the significance of most experiments. Figure 42 depicts main issues concerning benchmark samples including cardinality distribution, features per user distribution and rows per user of the proposed benchmark.

Figure 42 Benchmark statistical distribution



In addition to the statistical distribution provided in Figure 42, a complete description of the data in the benchmark is provided. Table 9 includes the characterization of the benchmark including number of users, number of features and main cardinalities regarding the data.

Table 9. Domain characterization

	Benchmark domain
Number of users	40000
Number of features	29
Maximum feature cardinality	53
Minimum feature cardinality	2
Average feature cardinality	7

Despite the fact that the benchmark has been obtained from a real domain research project, it has been normalized in order to make it suitable for any user model. However, as analysed during the state of the art some user models are created to perform over some particular domain. Those models might struggle when tested over this benchmark while that drop of performance is due to the model and not due to the benchmark. On the other hand, statistical user models have the ability to adapt to any domain (domain-independent) and therefore, will provide accurate results when performing over this benchmark. After the domain characterization provided in Table 9, a complete description of the samples is provided in Table 10.

Table 10. Samples characterization

	Benchmark domain
Number of users	$40 \cdot 10^3$
Maximum rows per user	33
Minimum rows per user	18
Average rows per user	25
Maximum features per user	27
Minimum features per user	18
Average features per user	22
Percentage of multi-valued features per user	3.33%
Percentage of multi-valued rows per user	5.42%

Once the training set (28000 samples) and the test set (12000 samples) were formed, a set of experiments is prepared. Thus, the benchmark contains experiments using the training set and experiments with the test set samples. In order to obtain these experiments a complete statistical analysis has been done. After this analysis, two confidence levels and two confidence intervals has been set for the benchmark. The proposed confidence levels are 95% and 99% while the confidence intervals are 2% and 5%. Combining these two values per metric provides different precision levels to meet any requirement. First set of experiments includes experiments through which a random fact is selected (to query the model) while the remaining facts are randomly ordered and fed into the model incrementally. On the other hand, the second set of experiments selects the inferable fact from a reduced subset of facts, alike the first battery, and the remaining facts are sorted and provided to the model. Table 11 shows main features of those batteries describing dataset.

Table 11. Data sets description

Data set	Confidence level	Confidence interval	Sample size
Training set	95%	2%	2250
Training set	95%	5%	400
Training set	99%	2%	3650
Training set	99%	5%	650
Test set	95%	2%	2050
Test set	95%	5%	400
Test set	99%	2%	3100
Test set	99%	5%	650

As previously mentioned, this benchmark proposes a solution to fill the lack of standardization and proper mechanisms for evaluate user models, therefore satisfying the needs of replicating evaluations and even supporting comparison of two different evaluations. In order to spread its use, it has been made available (under the link http://labda.inf.uc3m.es/doku.php?id=en:labda_lineas:um_1) and presented in specialized forums Calle, et al., (2013). By following that link, the reader can find the benchmark, documentation about the benchmark and some practical examples of use.

5.4 Evaluation Design

This chapter will go through some important aspects of the evaluation process including the preparation of the experiments, the models parameterization, and the load of experiments proposed to test every model. Before going deeper into the details of each single experiment, a complete description of the hardware and an overall parameterization of the models will be provided.

5.4.1 Experiments Preparation

Both proposals (RTUM and KLUM) have been developed using Java 1.6 and Pl/Sql. The knowledge bases are supported by Oracle 11g relational databases. All models rely on embedded Java 7 procedures. Those procedures are developed and inserted in the database in order to provide certain functionalities and improve response times. Having those procedures separate from the database involves network communication times that are usually opposed to bounded response times. Therefore, the best approach is to embed those procedures and make communications through Oracle 11g internal sockets that provide faster transfers. The experiments have been launched and tested using the in-house shell Cognos.User (Castaño et al., 2011). To perform the evaluation two Sun XFire with Intel Xeon E5450 clocked at 3GHz, 8GB RAM and SAS hard drives (4ms latency) are available. On the client side, a couple of Intel Pentium 4 clocked at 3GHz with 2GB RAM memory are running the software. All machines are running windows, on the server side Windows Server 2008 while the client machines are running Windows 7. Therefore, in order to replicate the experiments and provide the requirement of bounded response times those hardware specifications have to be replicated (same or higher hardware specifications). As important as the hardware requirements are the software specifications in order to keep the performance of the models within a reasonable margin. Next section covers the parameterization of the models along the experimentation.

5.4.2 Model's Parameterization

Regarding model's parameterization all models are set to provide bounded response times of less than one second (this parameter, which was a real research project requirement, has been taken for this evaluation, and is hereby proposed as common standard for future evaluations). In order to satisfy this requirement, some parameters have to be tuned. Thought, the response time depends not only on the model parameterization but also on the hardware resources. There are multiple parameters that will affect the performance of the models. In first place two parameters regarding the R-Tree structure will have a strong impact in the performance of the models. Those parameters are widely known as k and k_{min}. The value k represents the maximum population of any node in the tree. That is a maximum threshold that is going to trigger the partition process when exceeded. On the other hand, k_{min} is set to balance the nodes of the tree when distributing elements. The value of k for every model and every domain will be obtained through proper preliminary experimentation that has been specifically designed to provide information that reveals the most suitable value of k for each scenario, k_{min} values will be also obtained via preliminary experimentation. The value of k_{min} is usually fixed to 50 percent in the distribution of the B family trees. However, R-Trees trend to relax this value in order to provide better distribution despite the impact that it has on the tree structure. Thus, B, B+ and B* trees among others force k_{min} to be 50 percent in order to balance the structure of the tree. Fixing k_{min} to 50 percent assures a perfect balance when distributing elements between two nodes; therefore, the structure of the tree is completely balanced. On the other hand, that restriction might end up in non-optimal distributions. R-Trees used to relax that value accepting distributions with different balances, for instance 60 or 70 percent in order to guarantee the suitability of the distribution even when the structure of the tree can be good but not ideal. Thus, relaxing this criterion and keeping k_{min} values below 50% provides better distribution and also increases the success rate, while at the cost of unbalancing the sibling nodes load. Note that better distribution makes easier to select the proper branch of the tree and consequently increases the success rates. Furthermore, the structure of the groups of users within the tree is better and matching a new user with a group of users is easier and more accurate. This phenomenon will be reinforced through the evaluations over the leaves nodes of the tree. On the other hand, and as the main downside of this approach, relaxing the values of k_{min} means decreasing the minimum occupational density of the sibling nodes. The immediate concern of that reduced density is the growth of the amount of nodes, and consequently of the overall size of the tree. In general, lower density means larger size of the tree as their nodes might contain fewer elements. Note that the value of k_{min} has a strong impact in several parameters of the tree that will affect the performance of the model. Nevertheless, this fact has no real effect in the model's performance as long as the tree's depth is kept. Therefore the selection of both parameters k and k_{min} is crucial in order to maximize the performance of the models, and must be determined through preliminary experimentation.

Apart from k and k_{min} there are some other parameters that can be configured and will affect the performance of the model. The third parameter of the models is called the level of granularity and it refers to the specificity of the inferences performed over the model. This criterion allows configuring which condition will be used to go one level deeper during the inferences. There are multiple possibilities that can be developed in order to allow the model advance through the tree structure. One of the most common criterions is to fix a threshold. If that threshold is surpassed then the model will provide inferences in the next level. In fact, the models have been designed to accept thresholds in order to go down through the tree structure. Eight different thresholds have been tested in the models. Those thresholds are explained next.

The eight thresholds can be separated in terms of the success or failure of the prediction of the model. That is, four thresholds are measured when the model predictions' hit the expected value, and another four ones when the predictions are mistaken. Both sets of thresholds (success and failure) are equivalent. The first threshold is set to the certainty values measuring the minimum certainty to obtain a hit during the test or in the case of failure measuring the maximum certainty providing a fail. The second threshold has the same definition for the case of success and failure and its definition consists of measuring the iteration in order to estimate the best iteration to move down one level. The third threshold measures the difference between the first and the second candidate within the same iteration. In other words, through each iteration, the difference between first and second candidates is taken, in order to use that value to predict the best moment to go down one level. It should be recalled that, as described in the proposal, the best match has a child node endorsed, and that child node focuses a subset of the population described in the father node, with a more detailed description and higher certainty values, which will enhance predictions as long as the child choice is correct, but can lead to severe failure if the choice is mistaken. Finally, the fourth threshold measures the difference in terms of certainty between the first candidate of the iteration N and the first one of the iteration N-1. Therefore these are the eight thresholds that can be configured in the model. It is remarkable that the selection of one threshold or another will have a strong impact in the performance of the models. The thresholds are responsible for the granularity of the inference. Thus, at the beginning the models will provide more general inferences and ideally after several iterations those inferences should be more specific since they are supposed to be performed deeper along the tree structure.

The next parameter that can be set in the models is the inference method. All models developed admit to different inference methods in order to provide predictions. Despite both methods have been deeply explained in the proposal section of this document, a short explanation of this parameter is provided. The first method or maximum fit inference consists of taking the group with the maximum certainty. On the other hand, the second approach instead of taking the most similar group (best fit) looks for the second most similar group that usually has only slightly differences with the best one but may provide better inferences. Unlike the rest of the parameters explained, the inference method selection affects directly to the predictions and its value is crucial for the performance of the model. Similar to the inference method the partition method can be selected when configuring a model. Two main partition methods are available though some others might be developed, those methods perform the partition based on centroids or based on policies. Both approaches depends on the geometrical distribution of the users (items); however, the centroids method looks for the centroid of the groups while the privileged approach use to privilege one or more dimensions within that geometrical structure. An overall description of those mechanisms can be found along the proposal section of this document.

Another parameter that can be configured in the models is the 'fit method'. In present evaluation, this parameter has been set to the k-neighbours method. The k nearest neighbours is a well-known algorithm that measures the proximity of the input observations in the training set together with their related outputs in order to predict how close the objects in the test set are. It is noticeable that even when the models presented in this document only accept one value for the fit method parameter, this is still a parameter of the model and can be set to a different value just adding the implementation of another fit method to the model.

The two last parameters of the model are called the prune method and the update interval; and are responsible for removing useless data from the model and for setting the update frequency of the model respectively. The prune method parameter accepts two different values, prune using the number of rows within some particular node or prune using the standard deviation of all the rows within one node. As previously stated, one of the downsides of any knowledge model (and specifically, any user model) comes from the certainty degradation of the tuples within the model. Severely deteriorated tuples are not providing any information or knowledge to the inferences, while can make the system perform slower. A common solution to avoid this drawback consists of pruning tuples when they are so deteriorated that are deemed to be useless. Thus, one approach removes tuples when the number of rows within one node is higher than some threshold. On the other hand, the standard deviation approach removes rows when their standard deviation is far enough from the average to be sure that they have degraded enough. Finally, the update interval refers to how often the model is updated when a new user is inserted. This parameter has a strong impact in the efficiency of the model. When a new user is inserted in the model, the briefs in the parent's nodes all the way to the root of the tree become slightly outdated and a whole update is required. However, this small obsolescence won't have a noticeable impact on the predictions and can be delayed until some more users have been inserted in the model. The upside of this approach is the efficiency, fewer updates means better efficiency. Anyway, those updates cannot be postponed a big number of insertions because that might cause an impact on the predictions. Therefore, the values of this parameter are crucial for the efficiency and the accuracy of the model.

Once the definitions of the models in terms of their parameterization have been detailed, the chapter moves forward to detail the batteries of experiments and the workload of every model.

5.4.3 Workload and set of experiments

This section provides a whole run through the experiments proposed including those experiments within the preliminary experimentation and those within the evaluation phase. As aforementioned, the evaluation is performed over four different domains, however two domains are taken to perform the preliminary experiments. Those domains are: the real case domain selected to create the evaluation framework (from now on Benchmark domain) and the real domain obtained through the API of a well-known social network. The reasons to prefer these two domains over the other two are that these domains come from a real application case and both of them have been validated against an ontology. These reasons make those domains more suitable for the purpose of preparing the models for evaluation.

Even though, this section provides an overview of the sets of experiments, it starts with the preliminary experiments in order to present the parameterizations of the models that will be used during the rest of the evaluation. Right after the preliminary experiments the workload for each domain together with its peculiarities will be provided and analysed.

5.4.4 Preliminary Experiments

The preliminary experiments are aimed to provide relevant feedback about the most suitable parameterization of the model. The domain selected to complete the preliminary experiments is the Benchmark domain as it is specifically designed for user models and should provide a better feedback on the parameters. The set of analysed parameters and their focused ranges of valuation are displayed in Table 12.

Table 12 Preliminary Experiments

Model	RTUM	KLUM
K	5, 10, 20, 50	5, 10, 20, 50
K_{min}	30	-
Granularity	One out of four thresholds: Max/Min certainty, #Iteration, Diff_Same_Iter, Diff_Two_Iters	One out of four thresholds: Max/Min certainty, #Iteration, Diff_Same_Iter, Diff_Two_Iters
Inference Method	Fit vs Fit & Certainty	Fit vs Fit & Certainty
Partition Method	Centroids vs Privileged	-
Fit Method	Adapted K-Neighbours	Adapted K-Neighbours
Prune Method	#Rows	#Rows
Update Interval	Five insertions	Five insertions

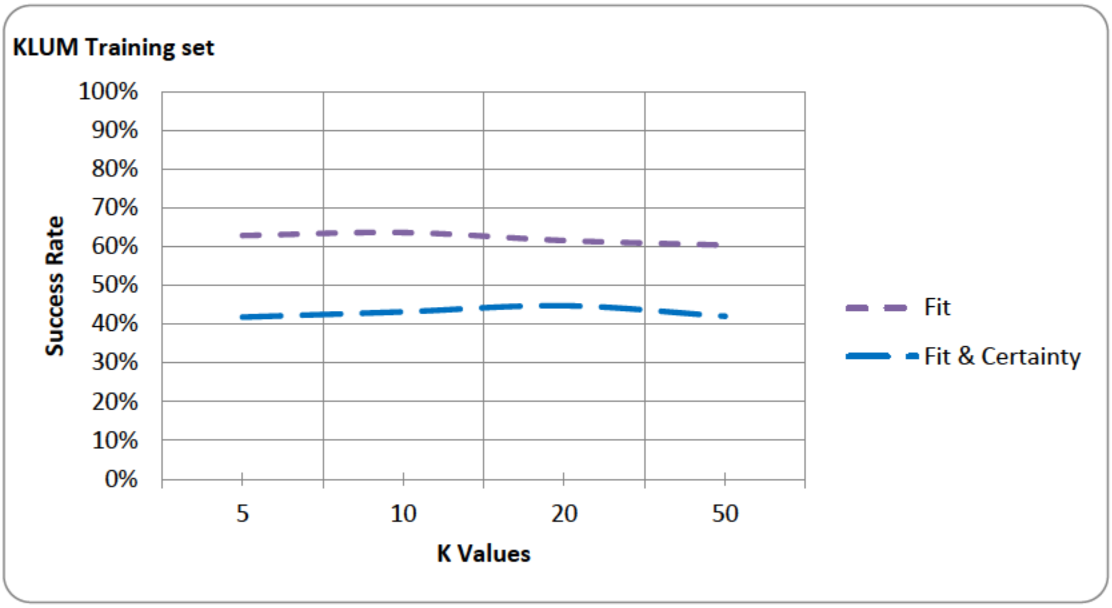
The first set of preliminary experiments tests both user models against the Benchmark domain. This experiment tests the fourth proposed values of K and the two suggested inference methods. Table 13 shows the results of this preliminary experimentation.

Table 13 Preliminary experiments results

	KLUM		RTUM	
K Values	Fit	Fit & Certainty	Fit	Fit & Certainty
5	62.87%	41.78%	70.10%	67.15%
10	63.65%	43.18%	70.77%	61.06%
20	61.55%	44.78%	70.93%	62.06%
50	60.34%	42.02%	71.58%	60.04%

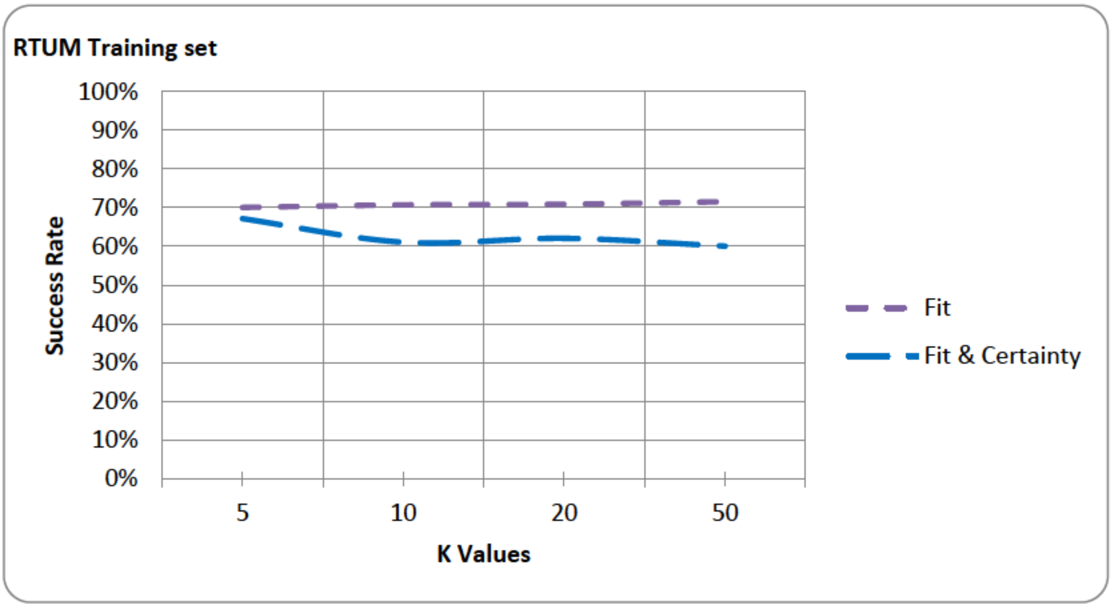
As can be seen on Table 13, the partition method of maximum fit always outperforms the fit and certainty. Irrespective, the value of K or the model the maximum fit method provides much better success rates than the fit and certainty. Regarding the models, RTUM outpaces KLUM in every single experiment although a comparison between both models is not the main purpose of the preliminary experiments. In order to support a better analysis of these results, the following set of charts depicts the former results. Figure 43 shows the aforementioned results for KLUM model on the training set.

Figure 43 Preliminary Experiments KLUM training set



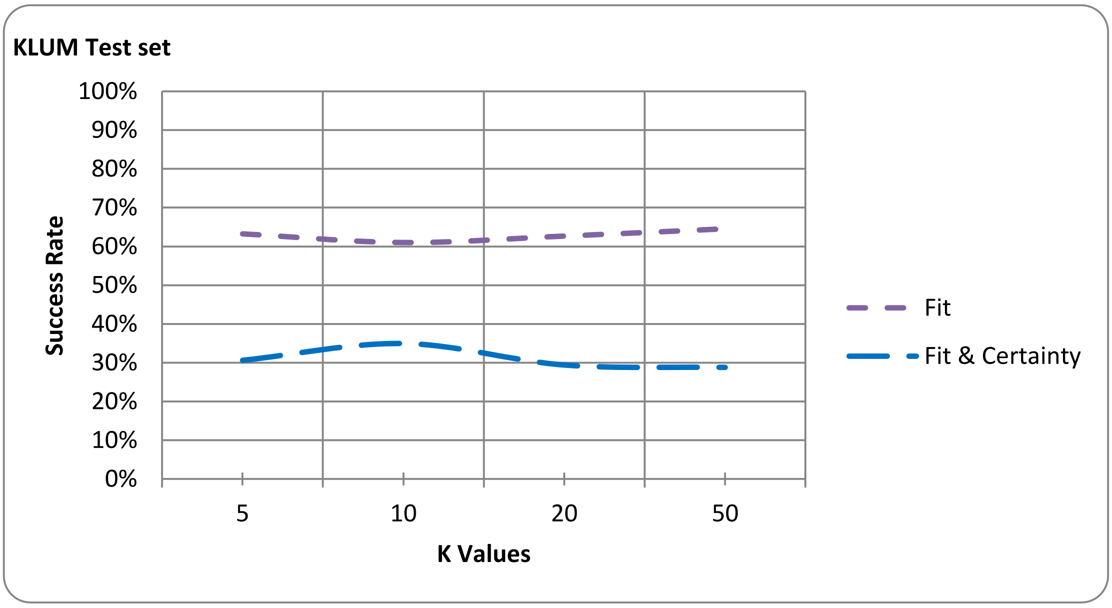
As previously mentioned, the fit partition method is much better than the fit and certainty, however in terms of values of K all produce a very similar result. Figure 44 shows the analogous experiment for model RTUM.

Figure 44 Preliminary Experiments RTUM training set



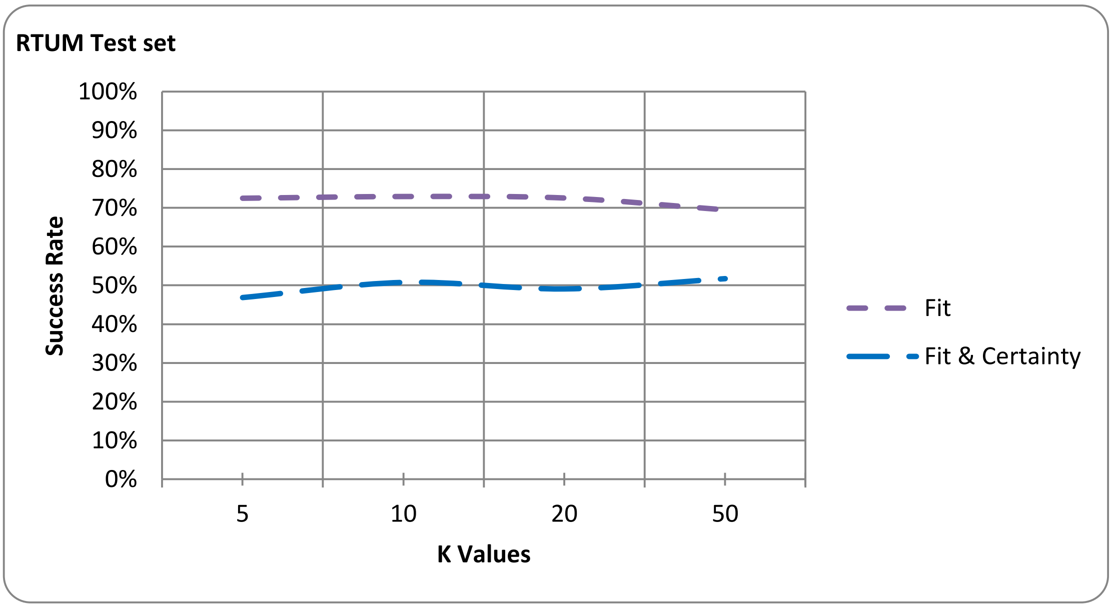
As shown in Figure 44 the fit partition model is once again better than the fit and certainty and it can be observed that attending to the results of the fit partition method is extremely difficult to pick one value of K. On the other hand, just looking at the fit and certainty partition method the smallest value of K seems to perform better than the others. After the experiments against the training set data sample, the next two charts cover the results obtained with the test set. Figure 45 has the results of the model KLUM evaluated against the tests set.

Figure 45 Preliminary Experiments KLUM test set



It can be seen that there is a massive gap between both partition methods being the fit method consistently better that the contender fit and certainty. As was the case with previous experiments, the best value of K depends on the partition method but even attending only to the best partition method it is difficult to make a decision. Finally, Figure 46 shows the equivalent experiment with RTUM performing over the test set.

Figure 46 Preliminary Experiments RTUM test set



The last experiment shows a not surprising gap between both partition methods and alike previous experiments the maximum fit method is the best. In addition, the line across different values of K is almost flat for this partition method, hardening the choice of the best K value.

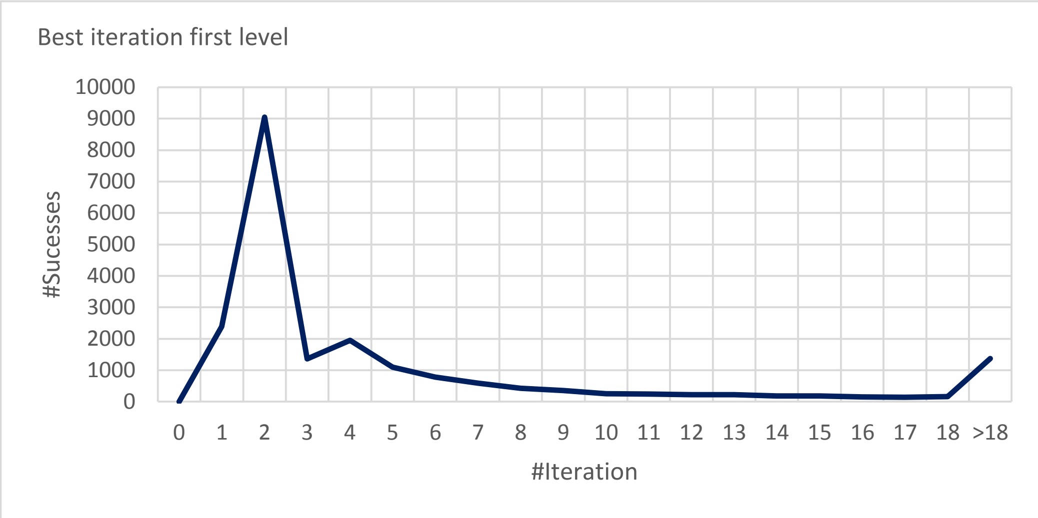
After the first series of experiments, a second set focuses on determine the best method to decide the granularity of the inferences. As explained in Section 5.4.2, this parameter refers to the specificity of the inferences performed over the model. This criterion allows configuring which condition will be used to go one level deeper during the inferences. Recalling again Section 5.4.2 of this document, those thresholds are calculated in two different sets. First set when the model hits the prediction and a second one when it fails. Those thresholds are only calculated from RTUM as this is the only model based on a tree structure. After the training phase, the resulting model has four levels and therefore three different thresholds have to be calculated in order to determine the best moment to go down from the root node to the first level and successively until ideally reaching the leaf level where there is more specific knowledge and the inferences should be more accurate. Table 14 shows the threshold calculations for the first level of RTUM according to the different criteria described in section 5.4.2.

Table 14 Thresholds first level

Certainty to go down	[<0.5]	[0.5-0.6]	[0.6-0.7]	[0.7-0.8]	[0.8-0.9]	[=1]
Minimum certainty hit	555	5551	7619	21	0	0
Maximum certainty fail	0	3311	174	0	0	15958
Best iteration						
Difference top-2 candidates hit	0	2206	37	0	0	16795
Difference top-2 candidates fail	556	5185	7582	21	0	0
Difference top-2 candidates hit (iter-1)	0	2355	43	284	171	81
Difference top-2 candidates fail (iter-1)	0	722	1968	451	253	93

Finally, due to the big number of iterations, the so called best iteration threshold is calculated according to Figure 47.

Figure 47 Best iteration threshold first level



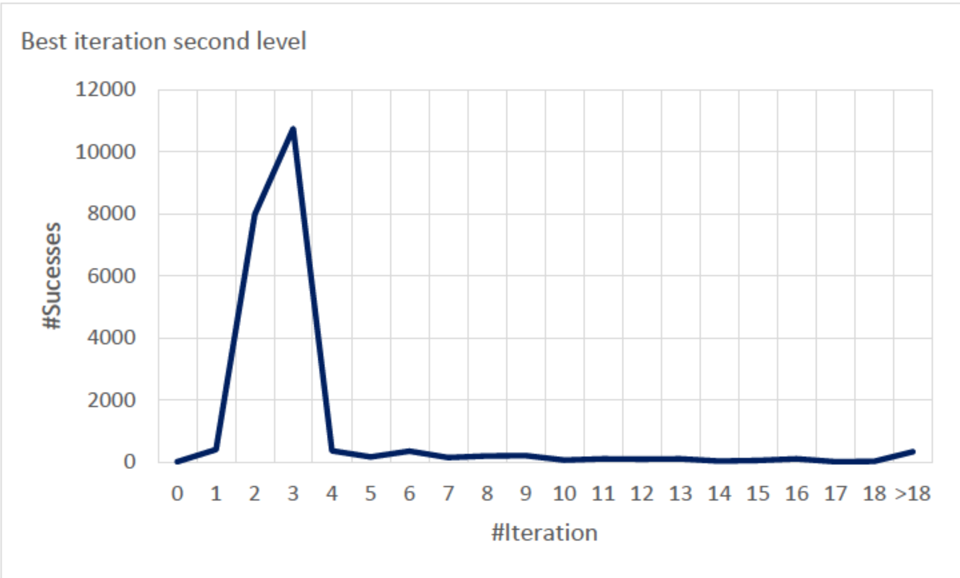
After the above calculations, one threshold has to be picked for the model, according to Table 14 and Figure 47 there are three candidates to be selected as threshold. Firstly, the minimum certainty of a hit seems to perform relatively well when this threshold is set to a certainty within [0.5-0.6]. Secondly, the difference of the top-2 candidates when failing the prediction performs only slightly worse than the first candidate. Finally, the best iteration method seems to be a clear winner as the second iteration can be set as a first level threshold achieving during the preliminary experiments a number of successes slightly over nine thousand. This threshold is then preferred and will be used for further experimentation. It is noticeable that setting the certainty to its maximum level will provide a very strong performance (15958 successes) using for instance the maximum certainty when failing the prediction and even a stronger performance (16795 successes) if using the difference in certainty between the top-2 candidates. The reason why these thresholds are ditch down in favour of the second iteration threshold is because reaching the second iteration will occur much earlier during the inference than reaching a certainty of one (assured fact). As ideally going deeper through the tree provides higher accuracy the threshold has to find a balance between two main factors, the number of successes and the time when the thresholds is reached. Hence, combining these two factors the best iteration set to the second iteration is unquestionably the best threshold for this level of the tree. To continue with the preliminary experiments Table 15 shows the threshold results for the second level of the tree.

Table 15 Thresholds second level

Certainty to go down	[<0.5]	[0.5-0.6]	[0.6-0.7]	[0.7-0.8]	[0.8-0.9]	[=1]
Minimum certainty hit	0	9408	253	0	0	7948
Maximum certainty fail	2386	7385	2183	0	0	0
Best iteration						
Difference top-2 candidates hit	0	7241	116	0	0	10256
Difference top-2 candidates fail	1036	6057	2164	1067	0	0
Difference top-2 candidates hit (iter-1)	0	5621	1347	1032	527	316
Difference top-2 candidates fail (iter-1)	0	4587	2023	950	467	90

Similarly to the previous experiments, the best iteration calculations are not included in the above table but are depicted in Figure 15.

Figure 48 Best iteration threshold second level



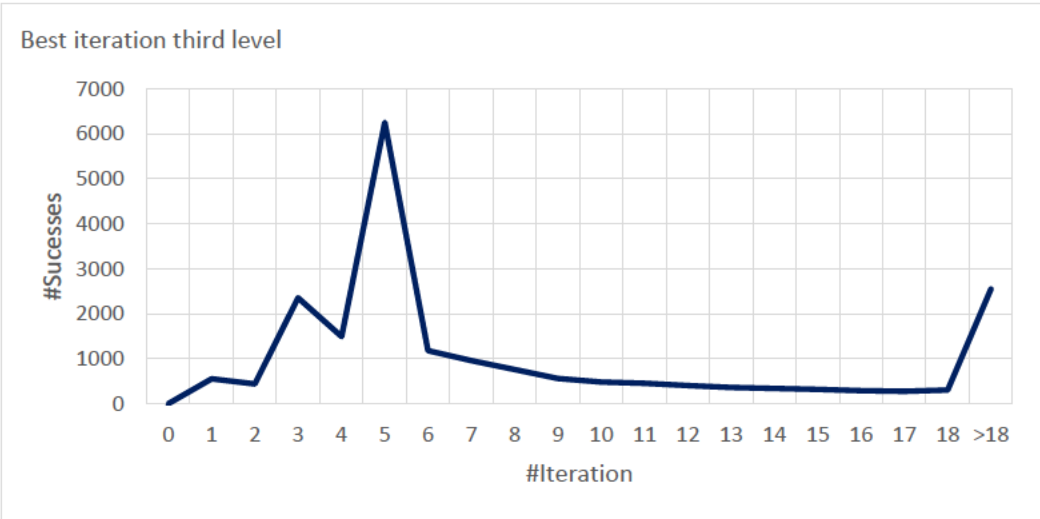
As can be seen in Table 15 and Figure 48 the thresholds for the second level of the tree are calculated similarly to the first level and the results offer little question about the best method for this new threshold. Even though setting the certainty to go down within [0.5-0.6] provides a big number of successes with at least three different methods, by setting the best iteration to three it is obtained the strongest performance, with more than ten thousand inferences correctly predicted. In addition, as this iteration is reached at the very early stages on the inference this method will be picked and used for further experimentation. Finally, Table 16 has the threshold calculations for the third level of the tree.

Table 16 Thresholds third level

Certainty to go down	[<0.5]	[0.5-0.6]	[0.6-0.7]	[0.7-0.8]	[0.8-0.9]	[=1]
Minimum certainty hit	722	1582	4518	4915	2115	0
Maximum certainty fail	0	2068	556	92	4	16206
Best iteration						
Difference top-2 candidates hit	722	2542	893	5280	4450	0
Difference top-2 candidates fail	0	1654	293	17	2	16795
Difference top-2 candidates hit (iter-1)	0	1335	4347	4884	2116	6764
Difference top-2 candidates fail (iter-1)	0	1968	451	253	5133	93

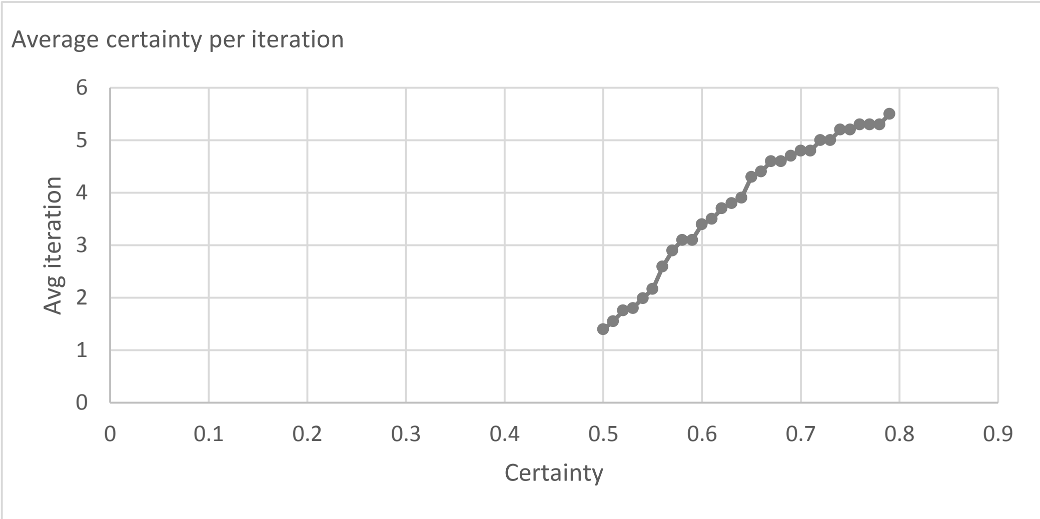
Analogous to previous threshold calculations, Figure 49 has the results for the best iteration method.

Figure 49 Best iteration threshold third level



Following Table 16 and Figure 49, two main candidates can be identified to be the third threshold. The first candidate is the difference between the top-2 candidates when hitting the prediction that performs strongly (5280 hits) when the certainty range is within [0.7-0.8]. However, attending only to the number of hits the best iteration method is actually better (6253 hits) on the fifth iteration. A deeper analysis is required in this case as the decision cannot only be based on the number of hits given that reaching a certainty within [0.7-0.8] might happen at an early stage and as previously mentioned going to deeper levels of the tree earlier is usually preferred. Hence, to make a decision between both candidates it is necessary to know how the certainty evolves through the iterations. Figure 50 shows the evolution of the certainty values provided with the iteration running.

Figure 50 Certainty versus iteration third level



As can be seen in Figure 50, a certainty of 0.7 is reached on average before the fifth iteration but between the fourth and the fifth and therefore the difference is not big as most of the times the model will go one level down between the fourth and the sixth iteration which is actually similar to using the best iteration set to fifth iteration and as this method provides better number of hits it will be preferred and the model will be set to reach the leaf level after the fifth iteration.

5.4.4.1.1 Preliminary Experiments Analysis and discussion

From the results of the preliminary experiments detailed above, some conclusions can be obtained. First of all, that the maximum fit inference method always performs better than the contender fit and certainty. Therefore, this inference method will be used for the rest of the experimentation. Second of all, it can be concluded that among the four K values tested (5, 10, 20, 50) is difficult to decide which one performs better as it seems to depend on the experiment and differences are very subtle across experiments. Also, it cannot be decided the best value of K for each data sample (training set, test set) as differences are again small and the best values in one experiment seem not to be the best in the next experiment. Even when this result suggests that any value of K can be acceptable for the experimentation, in order to have broader feedback, the values of K will iterate during the evaluation.

On the other hand, regarding the thresholds calculations, it can be concluded that the best iteration method outperforms the rest of thresholds tested. The best iteration allows the model to go deeper through the tree structure at an earlier stage and at the same time provides consistently the highest number of successes. Hence, this method is picked for the three thresholds as the preliminary experimentation shows this is the best method. The next section of this document provides a complete evaluation of the models in order to better understand their performance, weakness and strengths.

5.4.5 Evaluation

This section contains all the sets of experiments proposed to test the models in different scenarios. The section begins testing the RTUM user model parameterized per the previous section of this document and tested against the benchmark and social network domain. Table 17 summarizes the first set of experiments.

Table 17 RTUM experiments on benchmark and social domain

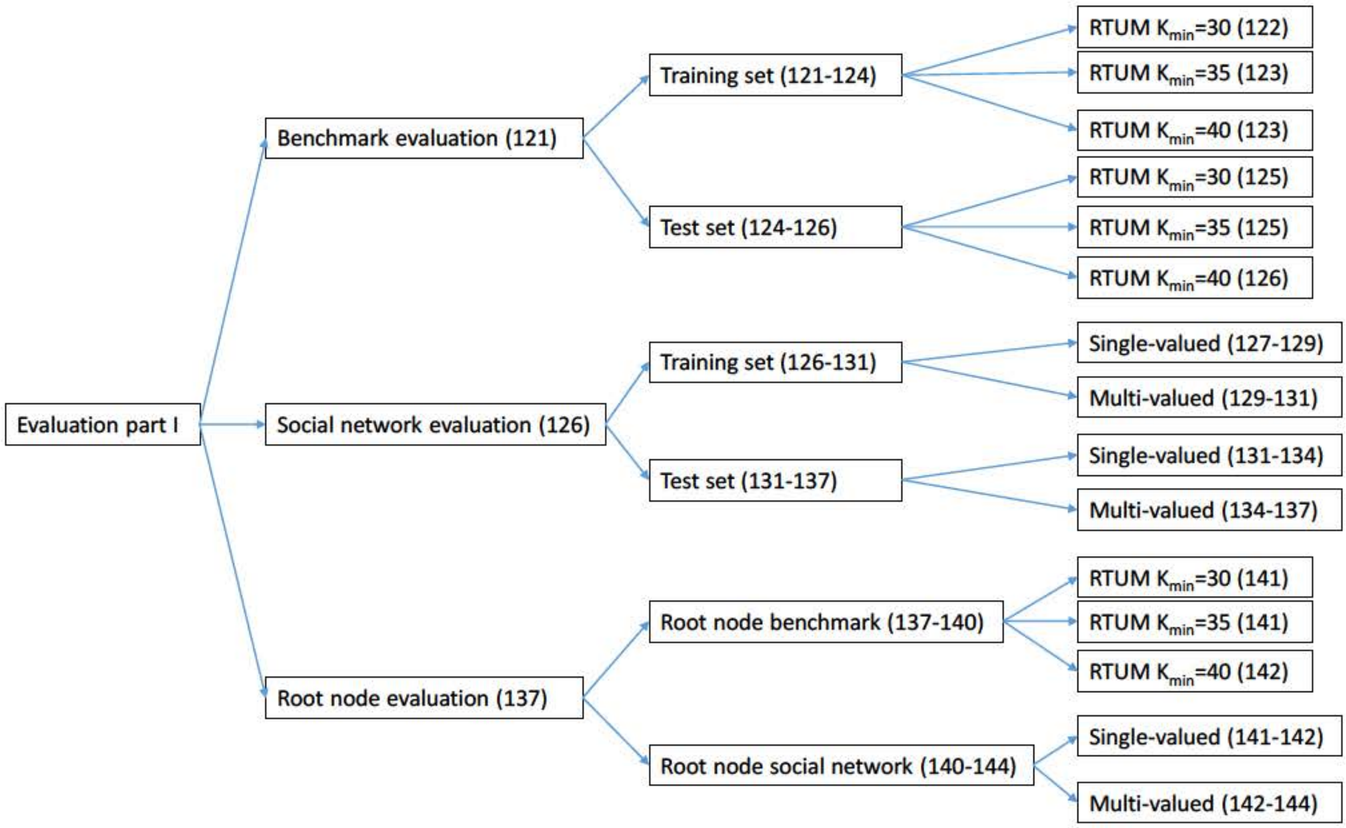
Domain	Benchmark	Social Network
Default Value	16%	0.45%
K	5, 10, 20, 50	5, 10, 15, 20
K_{min}	30,35,40	30,35,40
Granularity	#Iteration	#Iteration
Inference Method	Fit	Fit
Partition Method	Centroids vs Privileged	Centroids vs Privileged
Fit Method	Adapted K-Neighbours	Adapted K-Neighbours
Prune Method	#Rows	#Rows
Update Interval	Five insertions	Five insertions

As aforementioned, Table 17 splits the experiments depending on the domain. Although some parameters have been decided after the preliminary experiments (granularity and inference method), some others are iterated during this evaluation for completeness. Thus, the evaluation starts showing the results over the benchmark domain.

5.4.5.1 Evaluation walk-through

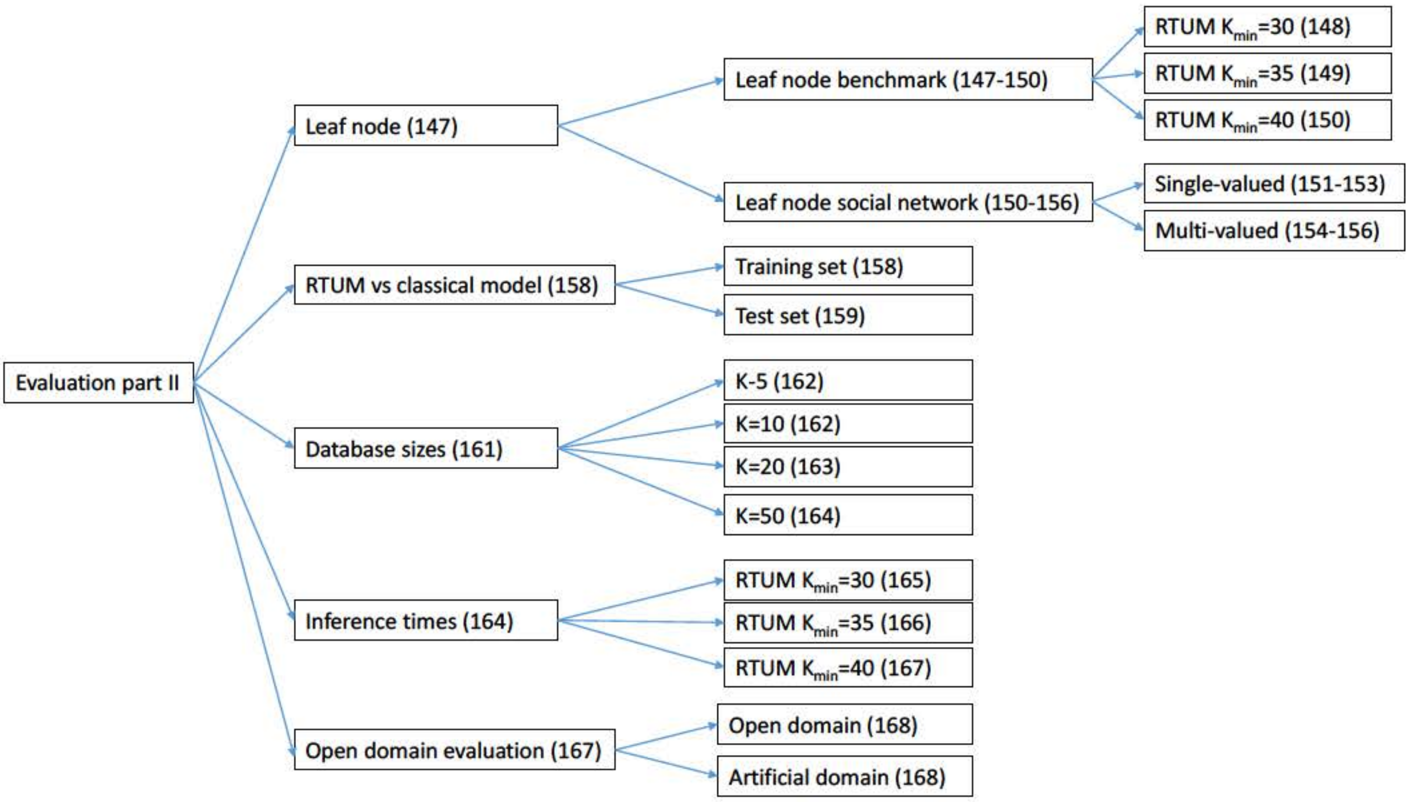
This section offers a quick guide to better understand and read the evaluation of this thesis. Due to the large amount of experiments, figures and tables, this guide helps to go through the evaluation or jumping into a subsection. Figure 51 contains the map of the first part of the evaluation.

Figure 51 Evaluation guide part I



The previous figure shows in brackets the page number of page range for each individual bit of the evaluation. Figure 52 contains the map to read the second part of the evaluation.

Figure 52 Evaluation guide part II



After this guide to the evaluation section, the rest of the chapter contains the evaluation of the models according to the above figures.

5.4.5.2 Benchmark domain evaluation

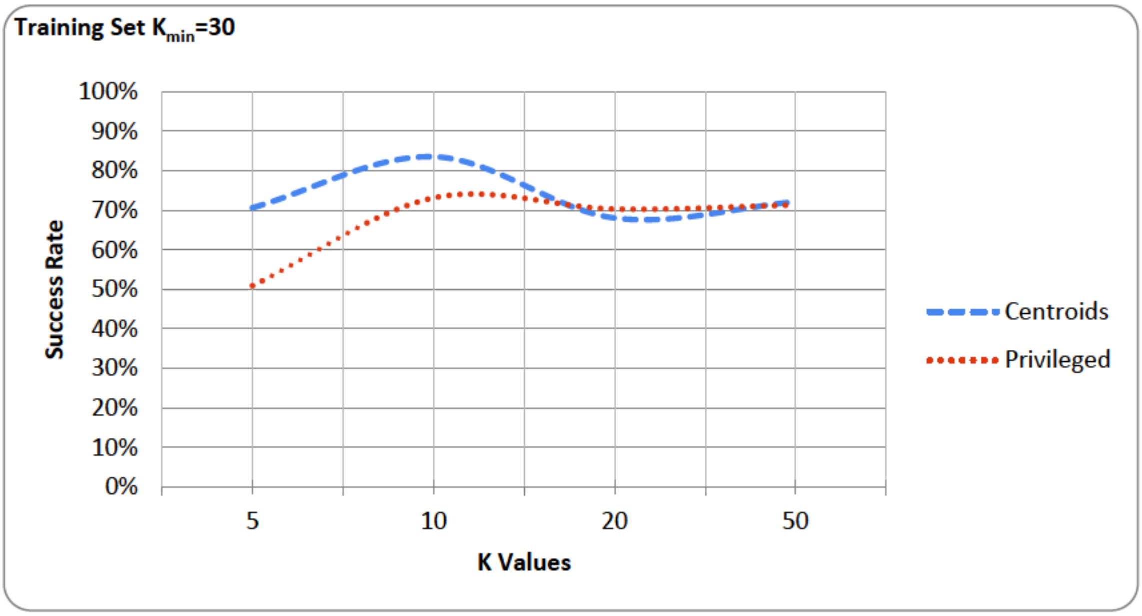
This set of experiments test the so-called RTUM model against the benchmark domain. As previously stated, some parameters will iterate during this evaluation. Hence, the values of K and K_{min} will iterate during the experiments and also two different partition methods are tested (centroids and privileged). The centroids partition method computes two centroids within the multi-dimensional distribution of the users to perform the partition. On the other hand, the privileged approach favours one or more dimensions during the partition. This first set of experiments will fix the value of K_{min} across experiments. Initially K_{min} is been set to 30 while the rest of parameters iterate. Table 18 depicts main parameterization of the model involved in the first preliminary experiments.

Table 18 RTUM evaluation benchmark training set

Domain	Benchmark
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	#Iteration
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Training Set

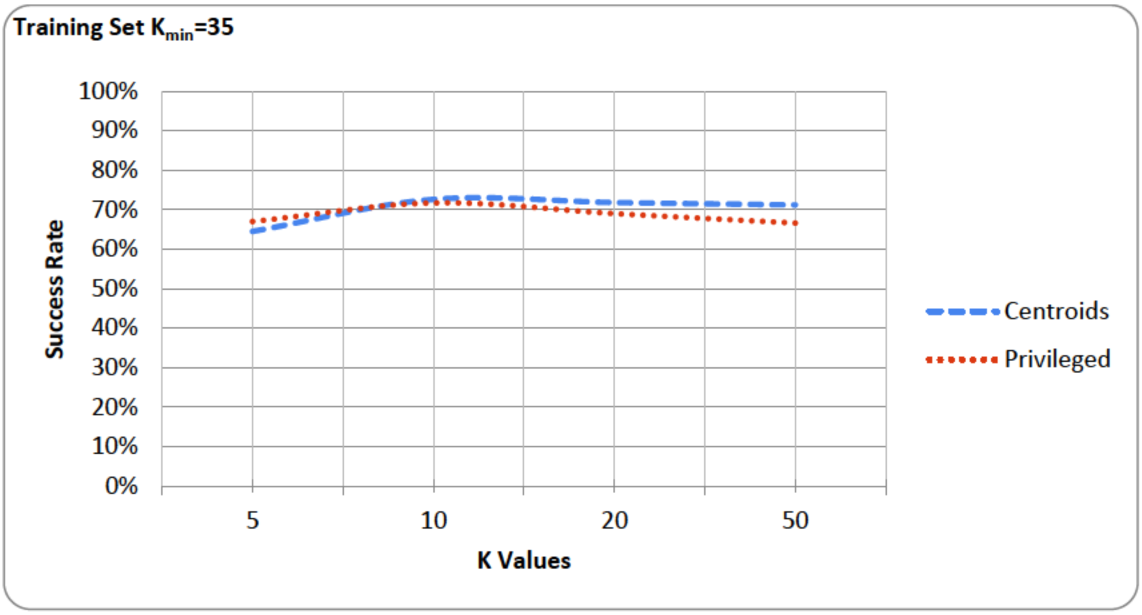
According to Table 18 the model called RTUM is being tested over the benchmark domain fixing the value of K_{min} to 30 while the values of K iterate. This experiment also allows a comparison between both proposed partition methods (centroids versus privileged features). Finally, Figure 53 shows the results yielded by the model when tested over a training set.

Figure 53 RTUM evaluation training set K_{min}=30



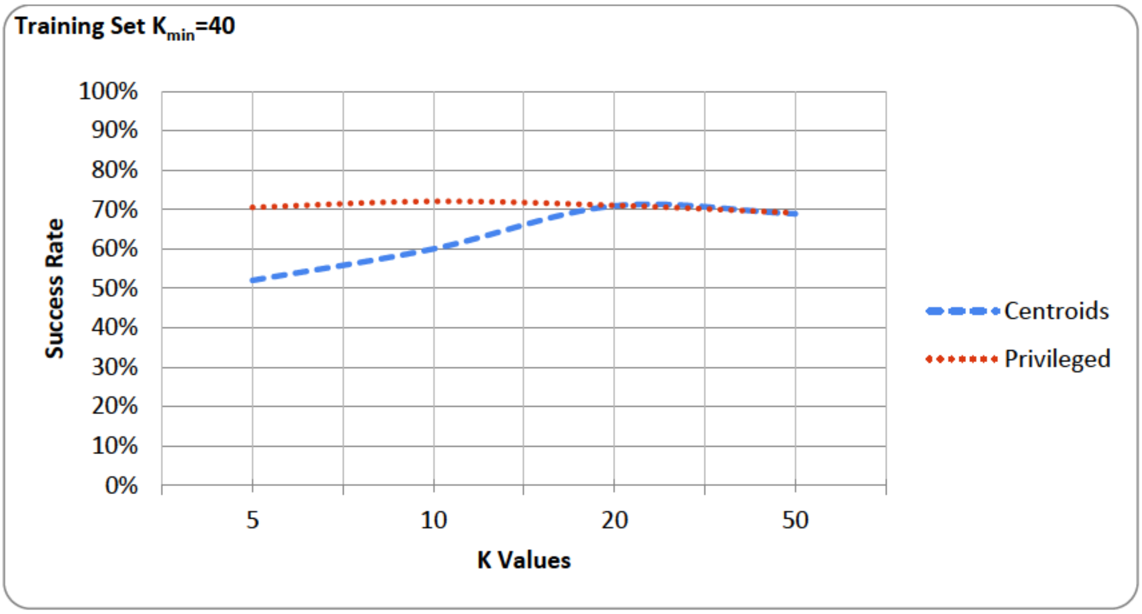
As can be seen in Figure 53, the model provides a remarkably high success rates (close to 85%) with the partition method of centroids providing generally better results. Attending to the performance of the model for the four different values of K, both models seem to reach their peak when k is 10. The next experiment proposed along the evaluation involves exactly the same parameterization except from the value of K_{min} that switches from 30 to 35. Figure 54 depicts main results obtained during this second experiment.

Figure 54 RTUM evaluation training set K_{min}=35



Alike in the first experiment, the centroids partition mechanism performs slightly bit better than the privileged method. In general, the behaviour of both methods is very similar, but again, both provide better results when k=10. However, unlike the first experiment the rest of data points in this second experiment are much closer to the pike. The third experiment of this series involves the same parameterization but tweaks the value of K_{min} to be set to 40.

Figure 55 RTUM evaluation training set K_{min}=40



The latest of this series of experiments shows a different trend where the privileged partition method performs better than the centroids method. In addition, there are two different pikes depending on the partition method, while the method of privileged features shows its peak again when K is 10 the method of centroids performs better when K is 20. Also the centroids method seems to struggle to reach its maximum when using low values of K. In terms of comparison among the different values of $K_{min,}$ the maximum value is reached when K_{min} is set to 30.

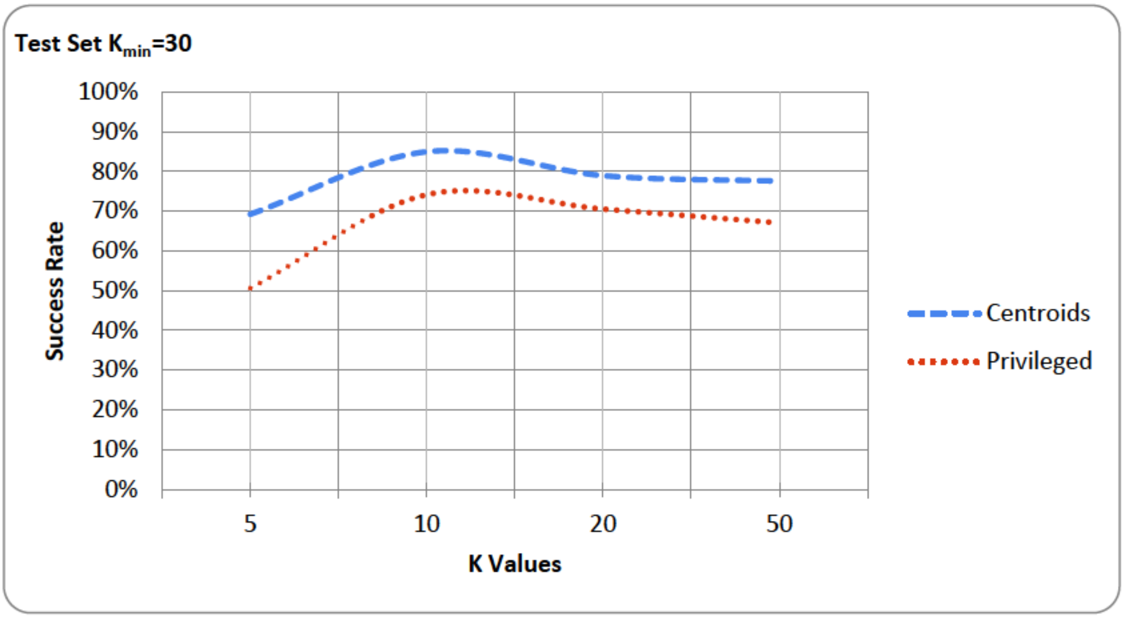
The first set of experiments has been performed over the training set of the benchmark domain, for the second series of experiments the parameterizations employed will remain the same whereas the experiments perform over the test set of the benchmark domain. Similarly to the first set of experiments, Table 19 depicts all parameters and models to be tested during this second series of experiments.

Table 19 RTUM evaluation benchmark test set

Domain	Benchmark
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	#Iteration
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

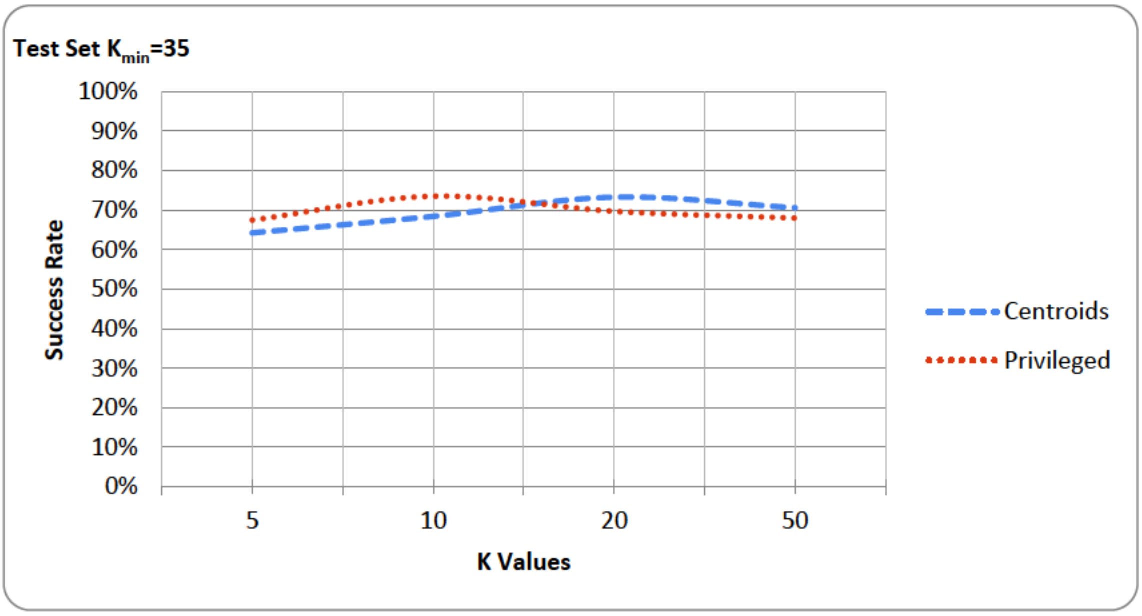
According to Table 19 RTUM user model is tested using similar parameterizations to the experiments above but over the test set. Experiments over the test set prove the ability of the model to generalize and recommend to other users different to those employed during the training set.

Figure 56 RTUM evaluation test set K_{min}=30



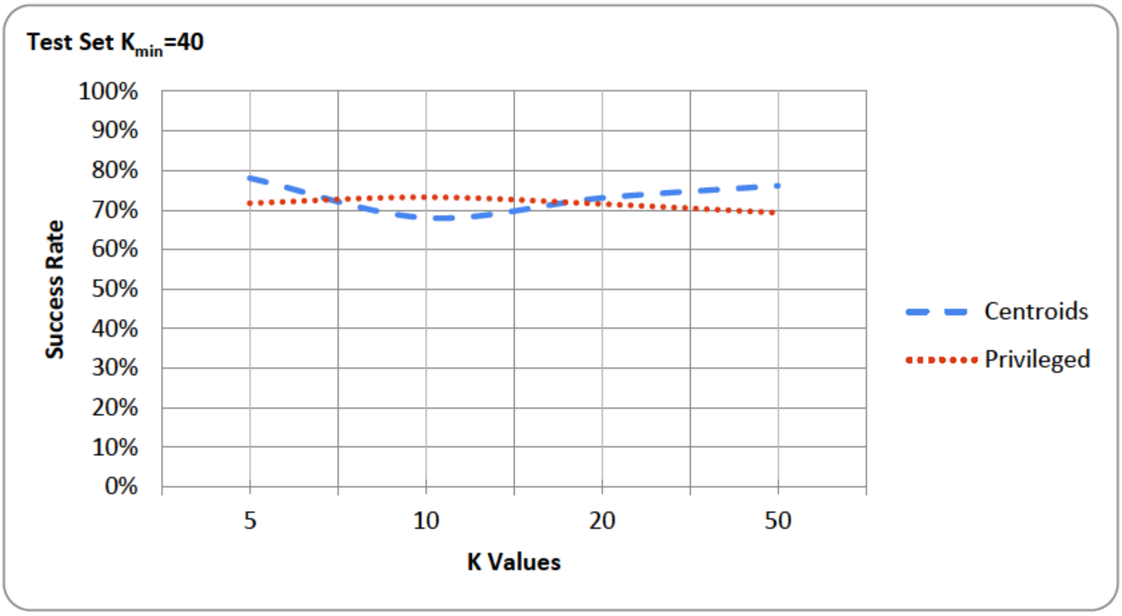
The first experiment (Figure 56) over the test set generates results where the centroids partition method provides better rates for all the values of K. Though, for smaller values of K (5, 10) the gap between two partition methods is bigger and close to 20% in success rate, this gap gets reduced for greater values of K. In fact, the same effect happens when training the model with the training set and setting the value of K to 30. Both charts have similar curves and in fact, the success rates reached are fairly similar. The next experiment over the test set flips the value of K_{min} from 30 to 35. Figure 57 shows the results for this second experiment.

Figure 57 RTUM evaluation test set K_{min}=35



The chart above shows how the model performs over the test set when the value of K_{min} is set to 35. Figure 54 and Figure 57 are again similar with better success rates for the privileged partition method when performing with lower values of K and better results for the centroid mechanism for the higher values of K. It means that bigger groups of users seem to favour the centroids method. Figure 48 shows how both models perform on the test set when K_{min} is set to 40.

Figure 58 RTUM evaluation test set K_{min}=40



In this case, there are some noticeable differences between Figure 55 and Figure 58, the most obvious difference is that the centroids partition method performs much more consistently over the test set while the analogous experiment over the training set shows how the partition method struggles for lower values of K. Regarding success rates, those over the test set are even higher than the rates obtained over the test set. After this set of experiment, the model will be tested against the social network domain.

5.4.5.3 Social network domain evaluation

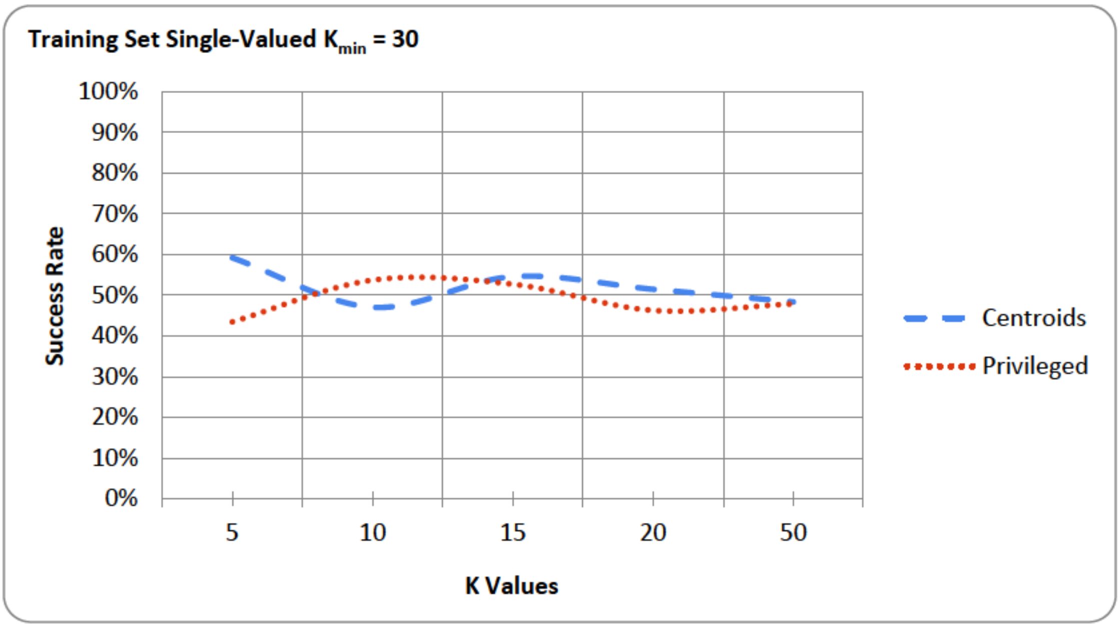
After the evaluation using the benchmark domain, analogue experiments are developed using the social network domain. This set of experiments involves fixing the value of K_{min} while iterating the values of K. Therefore, initially K_{min} is been set to 30 while the rest of parameters iterate. Table 20 displays main parameterization of the model used for these experiments.

Table 20 RTUM evaluation social network training set

Domain	Social Network
K	5, 10, 15, 20, 50
K_{min}	30, 35, 40
Granularity	#Iteration
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Training Set

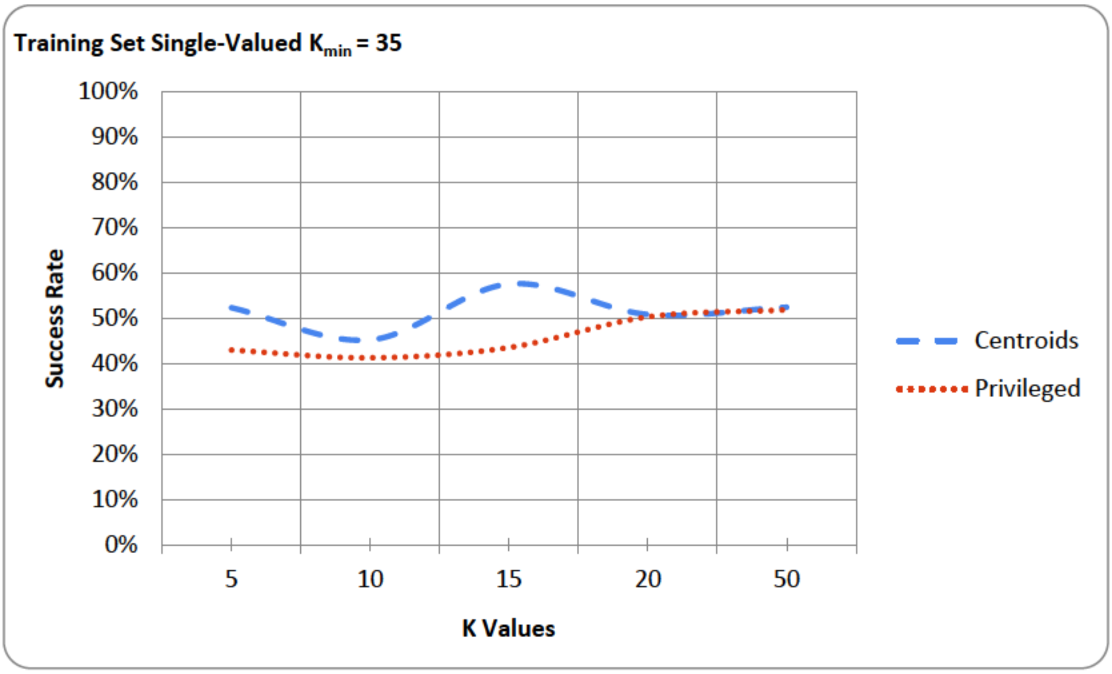
According to Table 20 the model called RTUM is being tested over the social network domain fixing the value of K_{min} to 30 while the values of K iterate. The model will be tested in two different scenarios regarding the nature of the features picked to make the prediction. The first round of experiments will use only single-valued features for the prediction whereas a second set of experiments will be based on predictions over multi-valued features. Figure 59 shows the results yielded by the model when tested over a training set and using single-valued features only.

Figure 59 RTUM evaluation training set single-valued K_{min} = 30



As can be seen in Figure 59 RTUM evaluation training set single-valued K_{min} = 30 the model based on centroids performs better than the model based on privileged features except when K is set to 10. The gap between both parameterizations is noticeable, yet not very big. As can be seen the results are lower than the analogous experiment over the benchmark domain where the centroids model provided success rates over 65% across different K values and a peak success rate of 83.5%. Lower results were expected due to the increased complexity of this model compared to the benchmark domain. The diversity of users and the special characterization of the domain makes difficult to reach those impressive success rates obtained with the benchmark domain. It is also remarkable that the default value (naïve algorithm outcome) of the benchmark domain is a significant 16% whereas the social domain has a much smaller 0.45%. The next experiment iterates the value of K_{min} from 30 to 35 to better understand the impact of this parameter on the social network domain. Figure 60 shows the results obtained during this experiment.

Figure 60 RTUM evaluation training set single-valued K_{min} = 35



Alike previous experiment, Figure 60 shows how the centroids partition method behaves regularly better than the privileged based sibling. Success rates are within the same range of the previous experiment but the peak success rate for this experiment is obtained when K is set to 15 whereas the previous experiment gets the best result when K is set to 5. The last experiment for the training set and using single-valued features is displayed in Figure 61.

Figure 61 RTUM evaluation training set single-valued K_{min} = 40

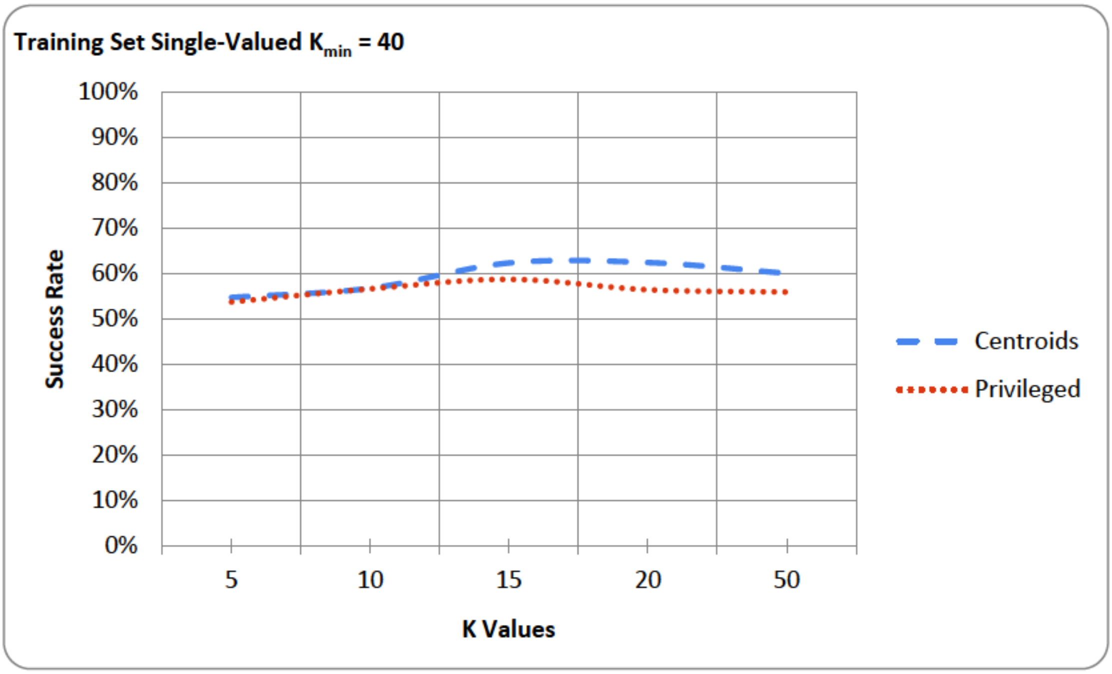
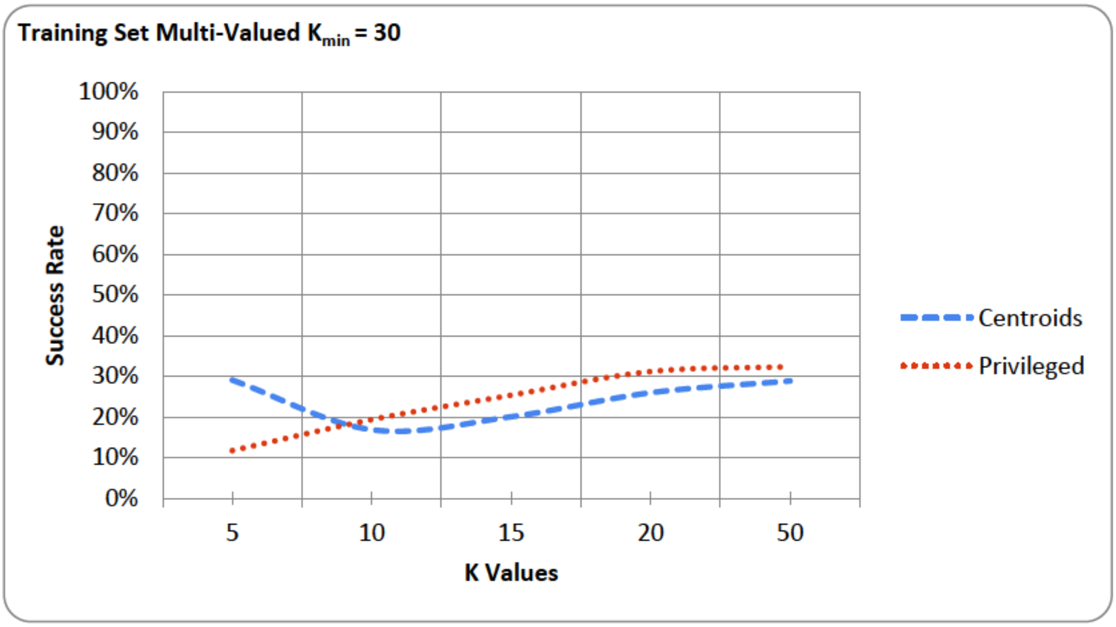


Figure 61 shows slightly better success rates than previous experiments within training set and single-valued features. Regarding partition methods, the model based on centroids once again performs better across K values and seems to have a more consistent behaviour over the three experiments. The following series of experiments offers the same experimentation over exactly the same dataset but changes the nature of the features queried from single-valued features to multi-valued features. Multi-valued features are inherently more difficult to predict and therefore the expected success rates are lower than those obtained for the models when dealing with single-valued features. Figure 62 shows the results of the first experiment where K has been set to 30.

Figure 62 RTUM evaluation training set multi-valued K_{min} = 30



As can be seen in Figure 62 the models provides good success rates but definitively lower numbers than those reached with single-valued features. It is also noticeable that for this particular experiment with multi-valued features the privileged partition method behaves marginally better than the centroids method for most of the K values. Next experiment of this set alters the value of K_{min} from 30 to 35 while the rest of the parameters remain the same. The outcome of the second experiment can be seen in Figure 63.

Figure 63 RTUM evaluation training set multi-valued K_{min}=35

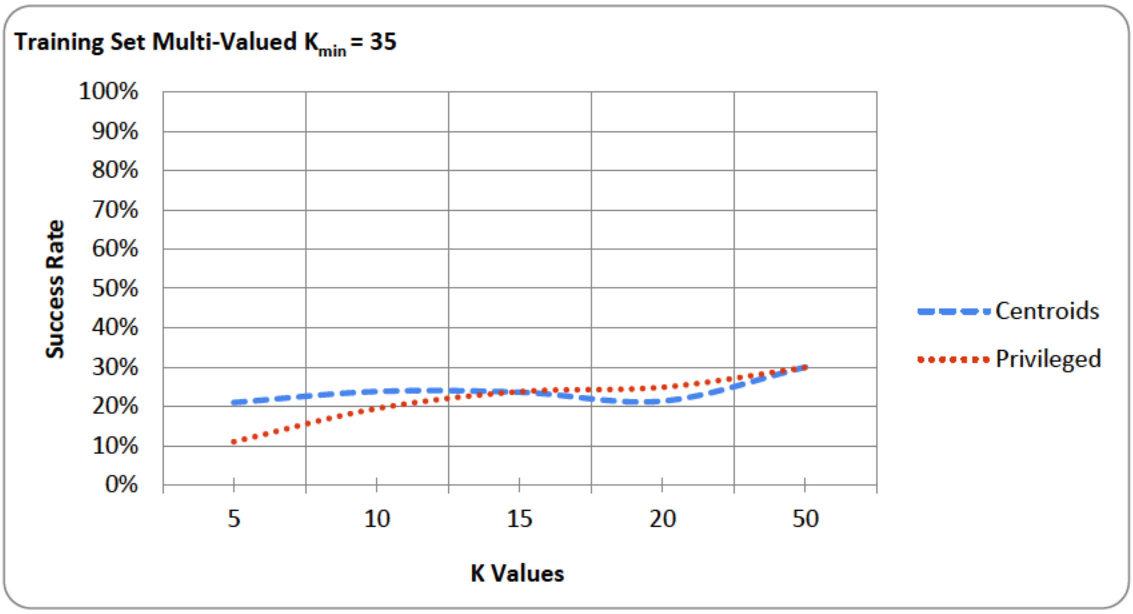


Figure 63 shows similar results when compared to the previous experiment. However, in this case unlike Figure 62 the privileged model is not always the best. In fact, the privileged model starts way behind the centroids model but manages to catch up when K=15, thereafter it performs better when K=20 and finally both models are neck and neck in the final data point. The last experiment of the training set data sample involves same models and parameterizations except from K_{min} value that as usual is incremented to 40. Figure 64 displays the result of this experiment.

Figure 64 RTUM evaluation training set multi-valued K_{min}=40

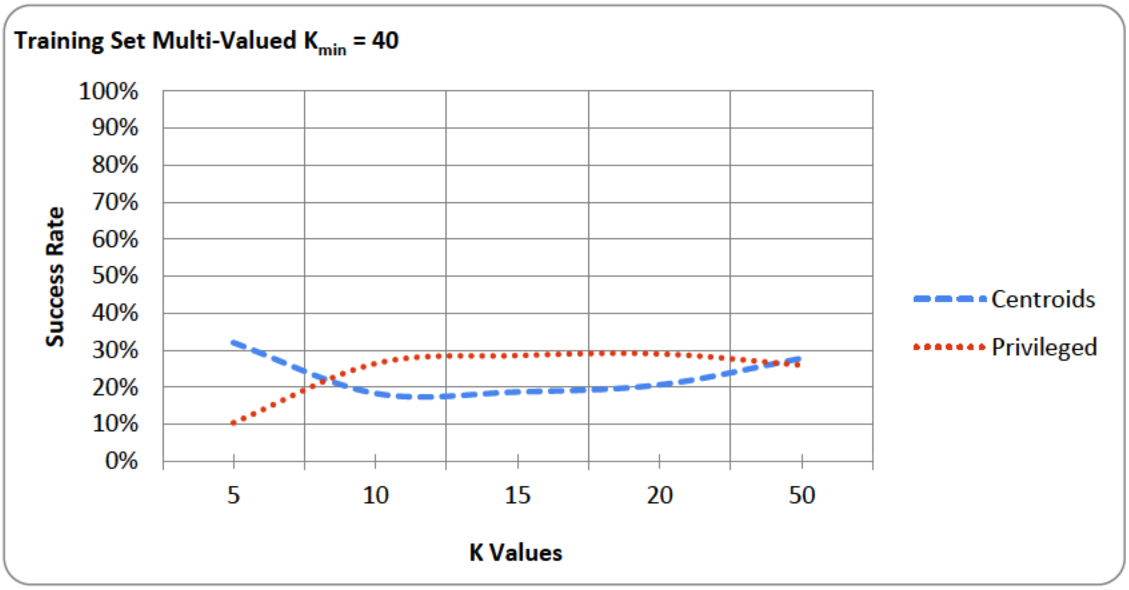


Figure 64 shows the last of this set of experiments over the training set and multi-valued features with K_{min} set to 40. Once again the values provided by the model range around 30% success rate and while the centroids method has the peak success rate the privileged seems to perform better for the majority of K values. In fact, interestingly enough the privileged method performs slightly better across this set of experiment with multi-valued features. To confirm these conclusions, the next set of experiments will assay the model against the test set data sample of the social network domain. Table 21 shows the experiments proposed for the test set.

Table 21 RTUM evaluation social network test set

Domain	Social Network
K	5,10,15,20
K_{min}	30, 35, 40
Granularity	#Iteration
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

As per Table 21 RTUM user model is tested again over the social network domain but in this case against the test data sample. First experiment fixes the value of K_{min} to 30 and iterates the values of K and similarly to the previous set of experiments, two different types of features are tested, first single-valued features and then multi-valued features. Figure 65 displays the result of the first experiment with single-valued features.

Figure 65 RTUM evaluation test set single-valued K_{min} = 30

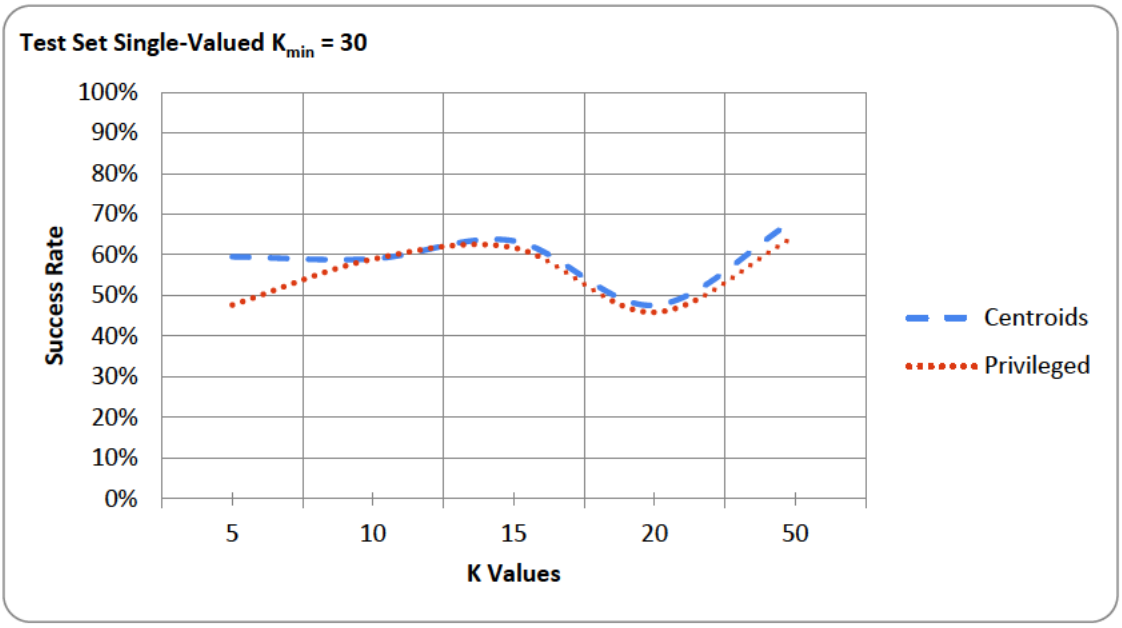
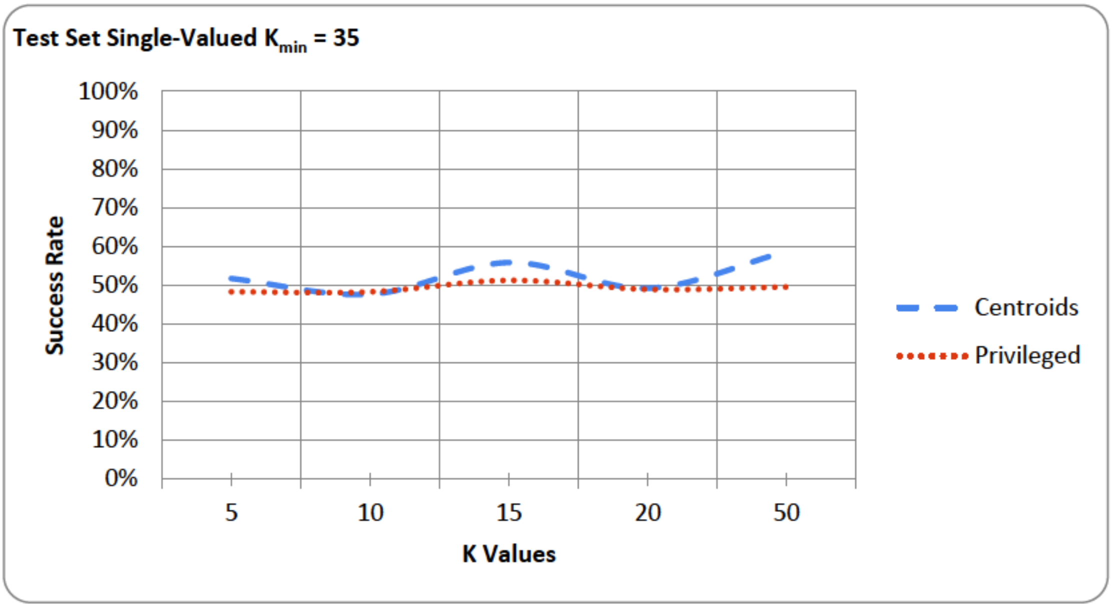


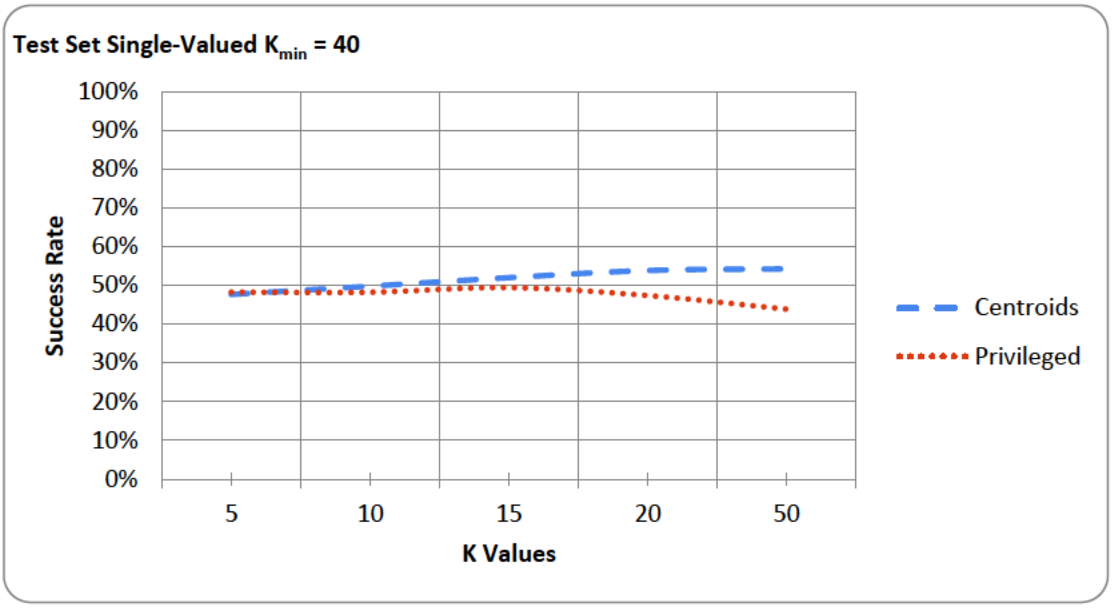
Figure 65 can be compared to Figure 59 (training set and single-valued features) where both models provided success rates under 60% and the average rates where around 50%. In this case, the peak success rate of both models is very close to 70% with the centroids model ahead and the average success rates are 59% for the centroids model and 55% for the privileged one. Therefore, the model generalizes quite well and shows a strong performance with the new data sample. Though both models yield very similar results the centroids model seems to behave better once the evaluation uses single-valued features again. Figure 66 shows the analogous experiment where the value of K_{min} has been increased to 35.

Figure 66 RTUM evaluation test set single-valued K_{min} = 35



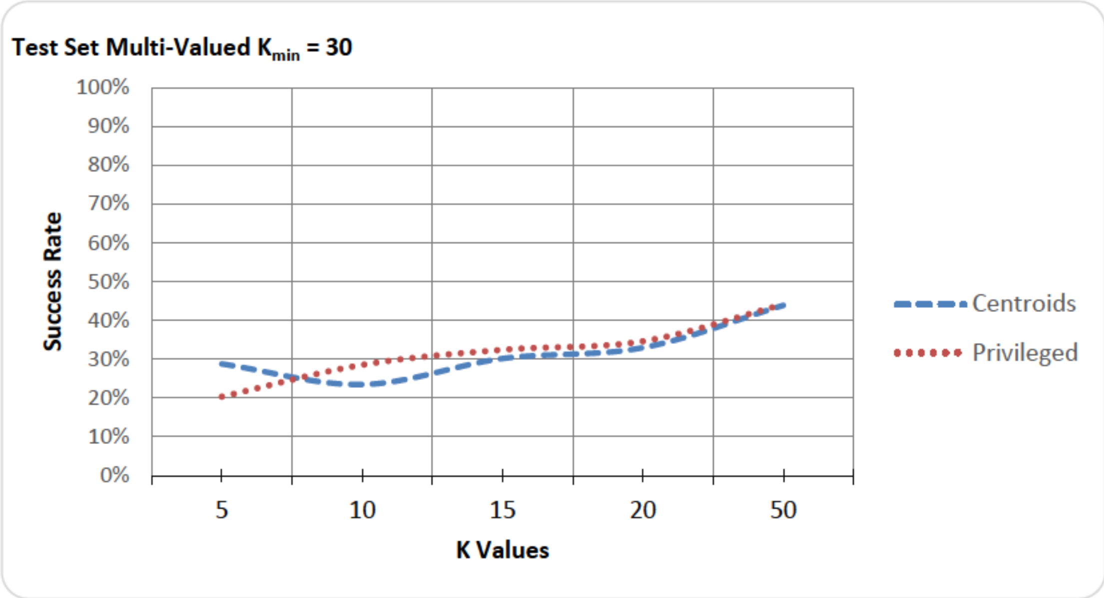
As can be seen in Figure 66 the centroids model performs somewhat better for most data points and significantly better when K is set to 50. In terms of data sample the results are comparable to those obtained using the training set (Figure 60) being the peak success rate for this experiment 1% higher than the analogous experiment. Finally, Figure 67 shows the experimentation for the highest value of K meaning allowing an unbalance of 60% within the tree partitions.

Figure 67 RTUM evaluation test set single-valued K_{min} = 40



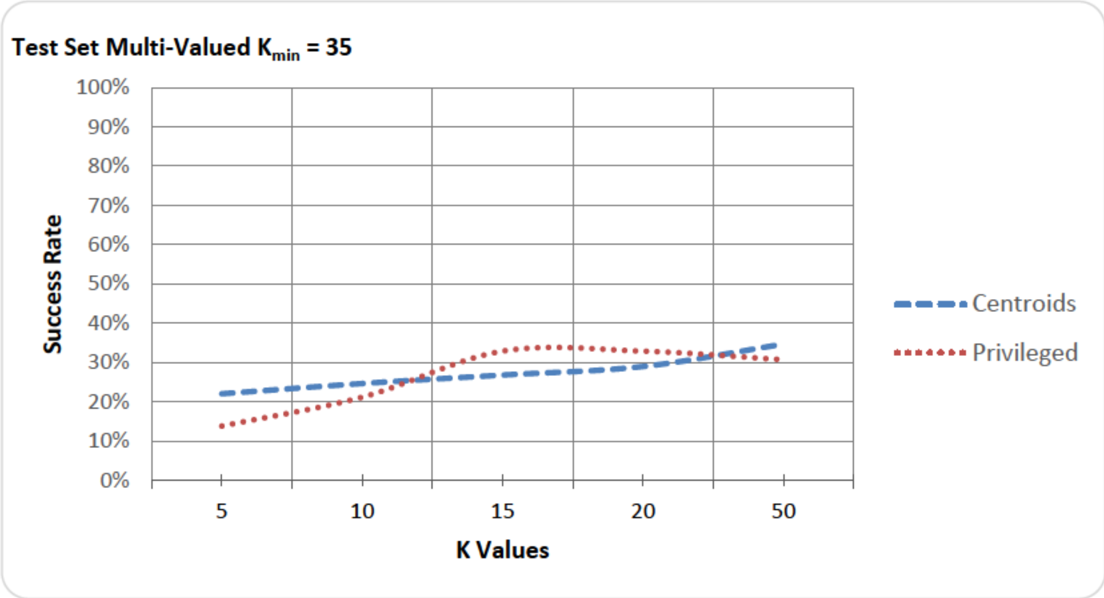
Alike previous experiments the centroids model outperforms the privileged especially for higher values of K. It is remarkable that the three experiments within this set obtained the best result when K is set to 50 and it is always the centroids model that performs better. The next three experiments test the model against the test set data sample and dealing with multi-valued features. It is clear that this final scenario is the trickiest one as involves generalization (test set data sample) plus multi-valued features that are inherently harder to predict. Figure 68 displays the first of those experiments for the lowest value of K_{min}.

Figure 68 RTUM evaluation test set multi-valued K_{min}=30



As shown in Figure 68, both methods of partition perform similarly, showing slightly bigger differences for lower values of K. However, when comparing the results obtained in the analogous experiment over the training set (Figure 62), it can be seen that the model performs significantly better over the test set. Both charts show a linear increase as the values of K increase thus, reaching their best success rate for the highest value of K. While this is not a concerning result, it is at least surprising and proves the ability of the model to generalize and provide accurate predictions out of the training sample. Thus, not only confirming the potential of the model but also validating the training phase. Next experiment just modifies the value of K_{min} from 30 to 35 thus allowing more flexibility at the time of distributing elements. Results are depicted in Figure 69 for both partition methods and different values of K.

Figure 69 RTUM evaluation test set multi-valued K_{min}=35



Once again Figure 69 shows a trend where both partition methods but specially the centroids based seem to improve their performance for higher values of K. Therefore, alike the previous experiment higher values of K produce better results whereas the model seems to struggle when parameterized with lower values of K. It is important to notice that K indicates the maximum number of groups allowed in each node of the tree. Poorer results for lower values of K might be associated to a premature partition of the node where most of the groups are very similar (given better results for higher values of K). This will vary across different domains and experiment though. Interestingly, the test set results are again slightly better than the corresponding over the training set just reconfirming the ability of the model to generalize. To finalize this set of experiments, Figure 70 shows the results for the highest proposed value ok K.

Figure 70 RTUM evaluation test set multi-valued K_{min}=40

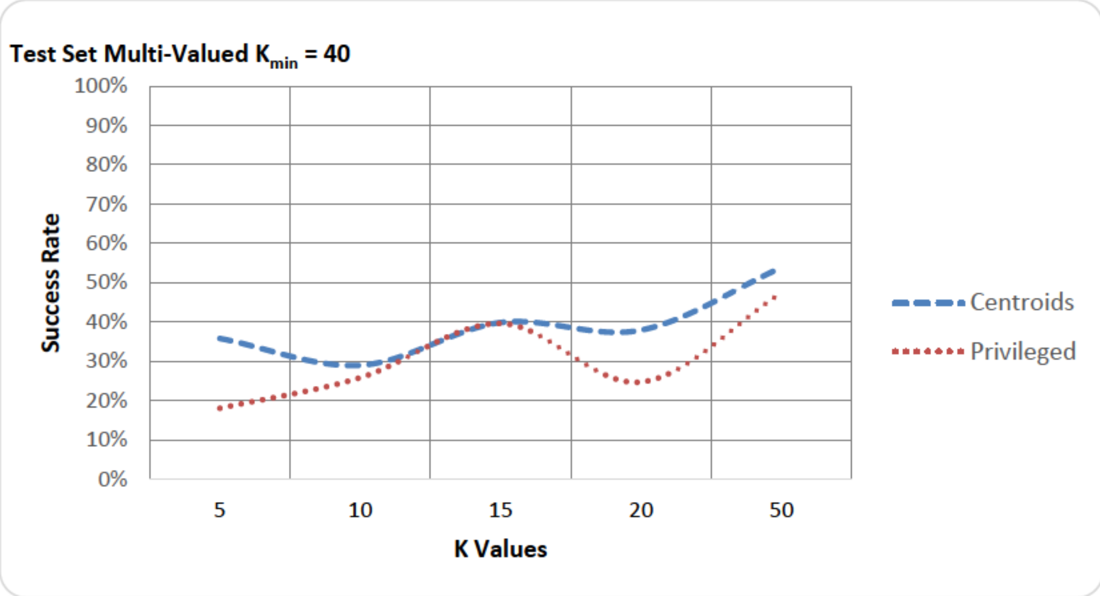


Figure 70, shows a similar behaviour for both partition methods although the centroid-based performs always better in this scenario. Similarly to previous experiments over the test set best results are obtained for higher values of K. After this set of experiments over the social network domain, the next series of proposed experiments will test RTUM model only over its root node. Section 5.4.5.4 has the experiments for the benchmark domain.

5.4.5.4 Root node evaluation benchmark domain

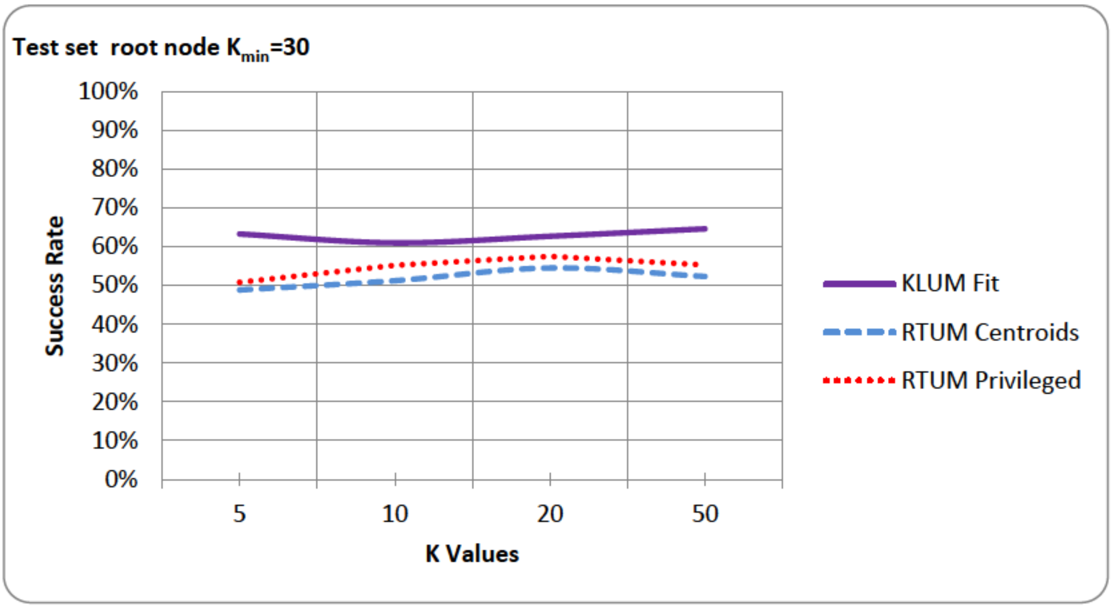
This section contains the evaluation of RTUM against the benchmark domain and using only the root node in the R-tree structure. There are two main reasons to do this evaluation: the first one is understanding how good is the performance of the root node compared to the whole tree; and the second one is compared the results of the root node with KLUM unique node which has been similarly created and evolved. During this experimentation, RTUM model is therefore not allowed to use its inference window and go to deeper levels of the tree to perform the inferences. Alike previous experimentation two partition methods will be tested in order to compare their performance. Table 22 shows the parameterization of the model employed for this experimentation.

Table 22 RTUM evaluation root-only test set

Domain	Benchmark
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	Root-only
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

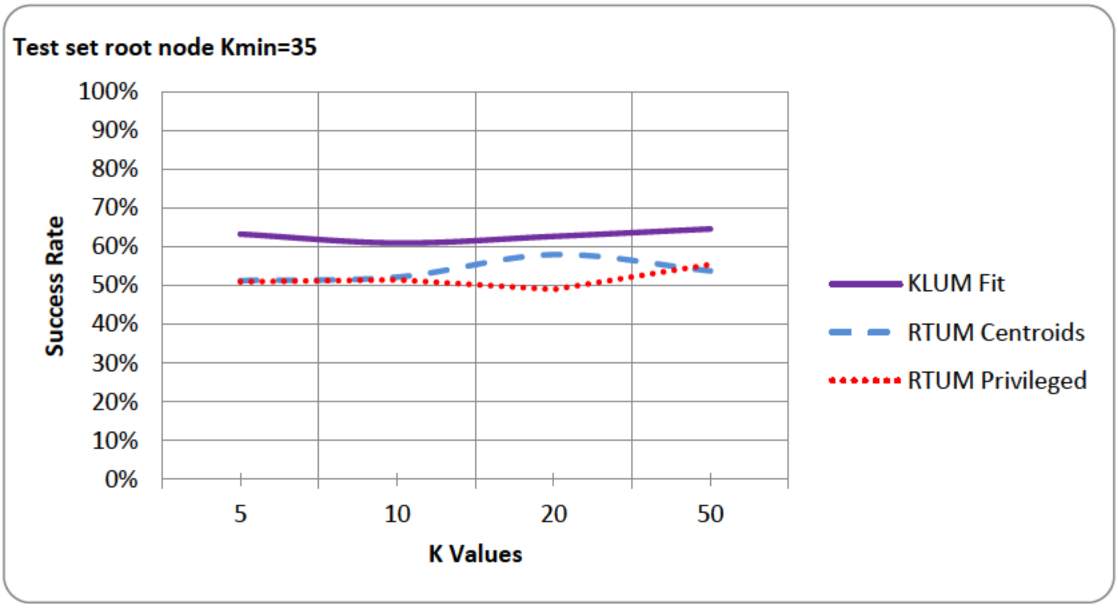
According to Table 22 the model called RTUM is tested against the benchmark domain and its granularity method has been set to root-only. Similar to previous evaluation, the values of K iterate for each proposed value of K_{min}. Figure 71 depicts the results collected during the first root experiment where K_{min} is set to 30.

Figure 71 Root node test set K_{min} = 30



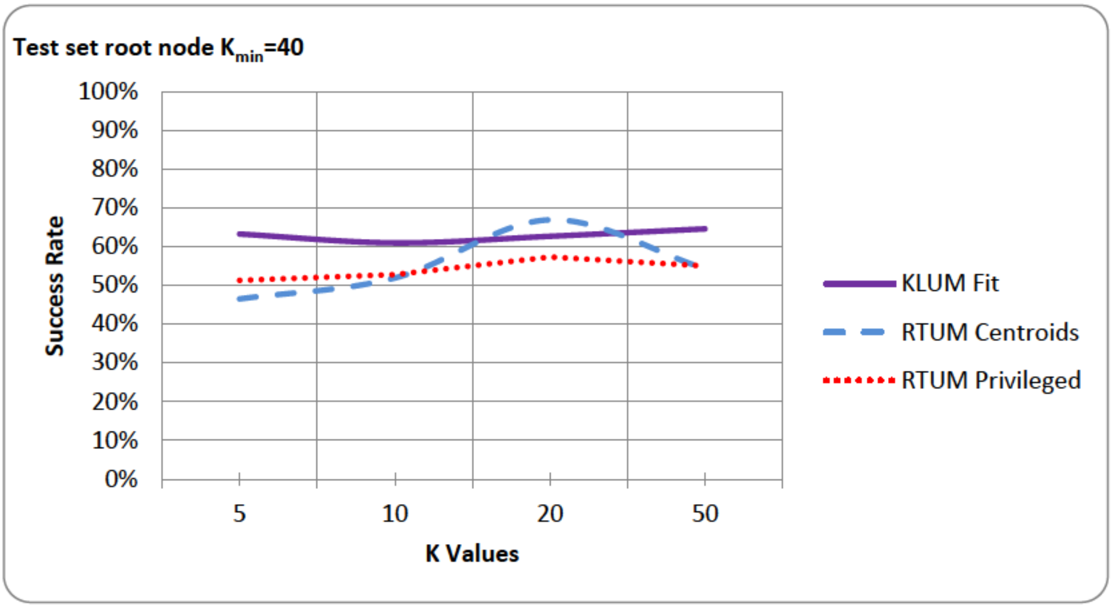
As the image above shows, both models have a very stable and similar behaviour being the model based on privileged features marginally better than the model based on centroids. It is remarkable that the analogous experiment considering the whole R-tree structure reached an impressive 84% success rate as can be seen in Figure 56. Consequently, this result proves the capacity of the model to use the inference window and perform more specific inferences and therefore provide much better success rates than the root node can provide. Regarding a comparison between KLUM (unique node) and RTUM root node, Figure 71 also shows the results of KLUM using the fit inference method (best result) for comparison purposes. It can be seen that KLUM achieves rates over 60% consistently and therefore, it can be stated that KLUM unique node outperformed RTUM for this particular experiment. Figure 72 has the results for the equivalent experiment where the value of K_{min} has been increased to 35.

Figure 72 Root node test set K_{min} = 35



As can be seen in Figure 72, once again the KLUM unique node seems to have a better clusterization than any of the RTUM proposed models. In fact, unlike Figure 71, the centroids model outperforms the privileged model for the majority of data points. Finally, Figure 73 shows the last root experiment for the benchmark domain where the value of K_{min} is set to 40.

Figure 73 Root node test set K_{min} = 40



Unlike previous experiments, the RTUM model based on centroids performs better than KLUM for one particular data point when K_{min} is set to 40 and K is 20. However, after these three experiments over the root node of RTUM model, it can be concluded that KLUM keeps performing better than the root node of the R-Tree based models. It is remarkable to note that in contrast with RTUM models, KLUM will not improve these rates as it is based in a unique node to perform the inferences. On the other hand, this is not a surprising result, since the KLUM is designed to work on a single node (a sole summary of all the population) while the R-Tree based models count on refined population to which dive for refining the results, and consequently are supposed to perform better once the root cap is removed.

After the root node experimentation on the benchmark domain similar experiments are proposed for the social network domain.

5.4.5.5 Root node evaluation social network domain

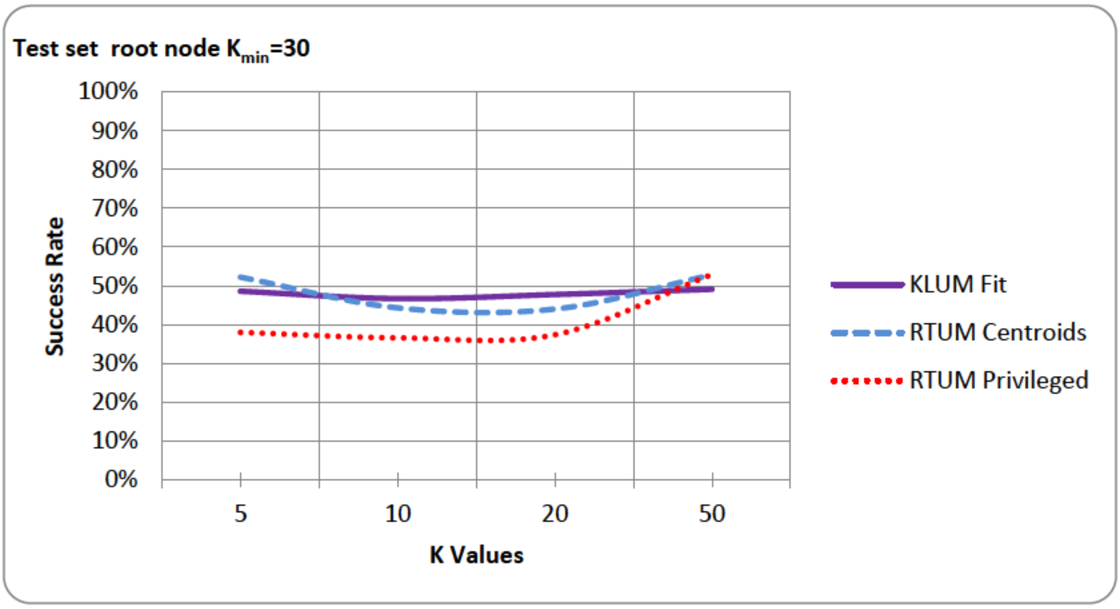
Alike previous experiments, this section contains the evaluation of RTUM model against the social network domain, for this experimentation RTUM will be capped to use only its root node and therefore asses the performance of the root node against the rest of the tree and against KLUM unique node. Table 23 shows the parameterization of the model employed for this experimentation. As previously stated the different values of K_{min} are only for RTUM as KLUM does not have this parameter due to its unique node structure.

Table 23 RTUM evaluation root-only test set

Domain	Social Network
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	Root-only
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

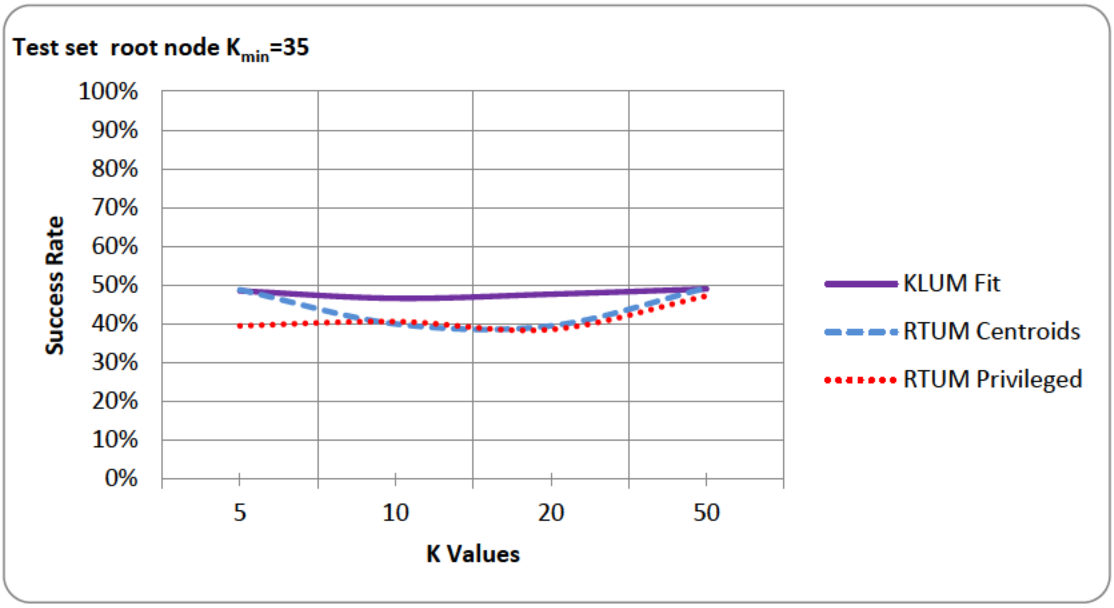
As stated by Table 23 RTUM user model is tested against the social network domain using its root node only, Figure 74 depicts the results collected during the first root experiment where K_{min} is set to 30.

Figure 74 Root node test set single-valued K_{min} = 30



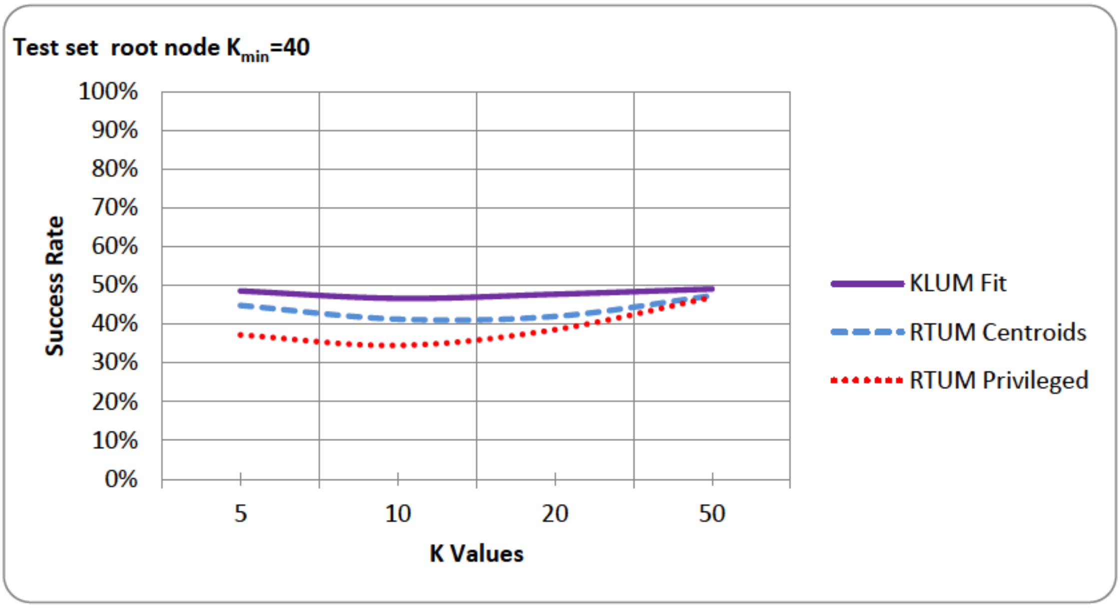
Similarly to previous experiment over the root node, the previous figure contains a comparison of RTUM model with two different partition methods and KLUM model with its best inference method. As shown in Figure 74, the so called KLUM model performs slightly better than the RTUM model for the central data points where RTUM seems to struggle. However, both models (KLUM and RTUM centroids) performed neck and neck along the experiment. As in the majority of experiments so far, the centroids partition method outpaces the privileged partition. Figure 65 includes equivalent experiment using single-valued features but removing the root cap for RTUM; results for the centroids model are on average a 10.35% better across different values of K while the privileged model has a 12.98% improvement when using its full structure. Figure 75 shows the next of this set of experiments where K_{min} jumps to 35.

Figure 75 Root node test set single-valued K_{min} = 35



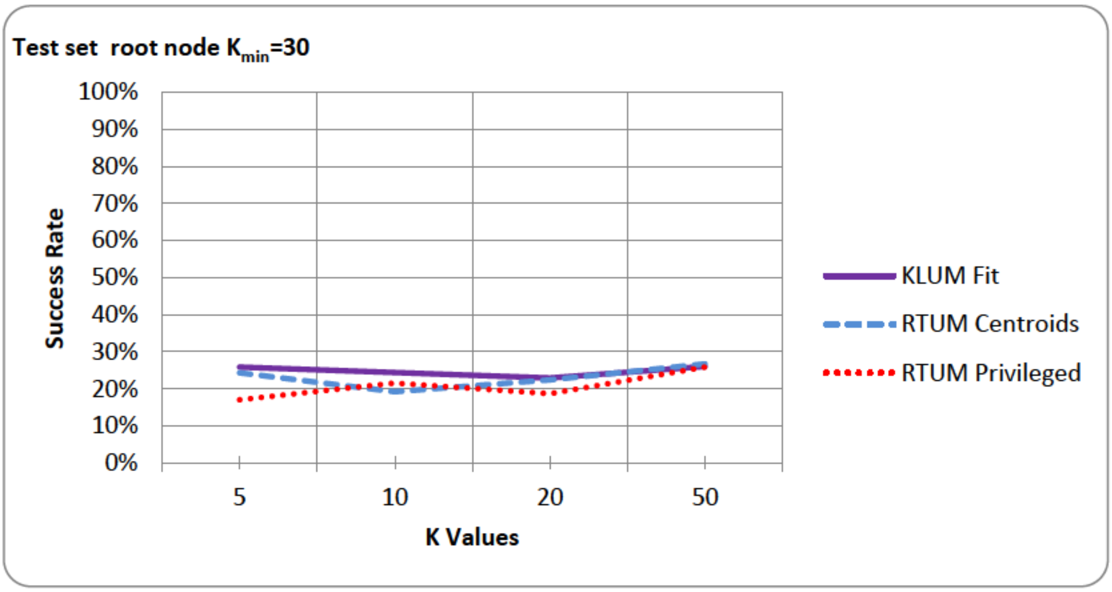
The comparison between Figure 75 and its analogous experiment in Figure 66 yields a significant improvement in success rate when RTUM uses its R-Tree structure. The centroids model achieves a 7.43% improvement for a 7.28% rise for the privileged model. As the above figure shows KLUM model is better than the root node of RTUM and the gap between both RTUM models is negligible. The reason why both RTUM models are so close in this experiment is basically due to the fact that the centroids model performs a bit worse when K_{min} is 35 and the privileged model does exactly the opposite. Thus, in this case the value, of K_{min} can have a different impact depending on the partition method. To confirm or deny this fact, a last experiment of this battery with K_{min} set to 40 is run, and its results shown in Figure 76.

Figure 76 Root node test set single-valued K_{min} = 40



The above image shows a similar picture to previous experiments over the root node where KLUM unique node performs marginally better than both RTUM models. Once again, the centroids partition method shows a better behaviour than the one with privileged features policy, although the gap between both models seems to vanish with bigger values of K. Also in comparison with the whole tree experiment, both RTUM models experience a 7.5% gain when using its whole R-Tree structure. Next, the same experiments are repeated but in this case dealing with multi-valued features. As previously mentioned, multi-valued features are harder to predict and this has an impact in the performance of the models. Figure 77 has the results of the first experiment.

Figure 77 Root node test set multi-valued K_{min} = 30



As seen in the previous chart, dealing with multi-valued features has a big impact in the performance of the models. Interestingly it seems to affect all models but the impact seems even bigger for KLUM that use to top the single-valued experiments and has seen its advantage reduced although it is still the best performer. In comparison with the single-valued experiments each model experiences a decrease in success rate with average dips between 20% and 25%. In terms of comparing the root node versus the whole tree, Figure 68 shows the experiment over the whole tree and the results are significantly better when using the inference window and the whole data structure. The centroids model performs better (9.05% better on average) while the privileged mode is on average an 11.11% better when allowed to use its full tree structure. Figure 78 shows the next proposed experiment where the value of K_{min} is increased to 35.

Figure 78 Root node test set multi-valued K_{min} = 35

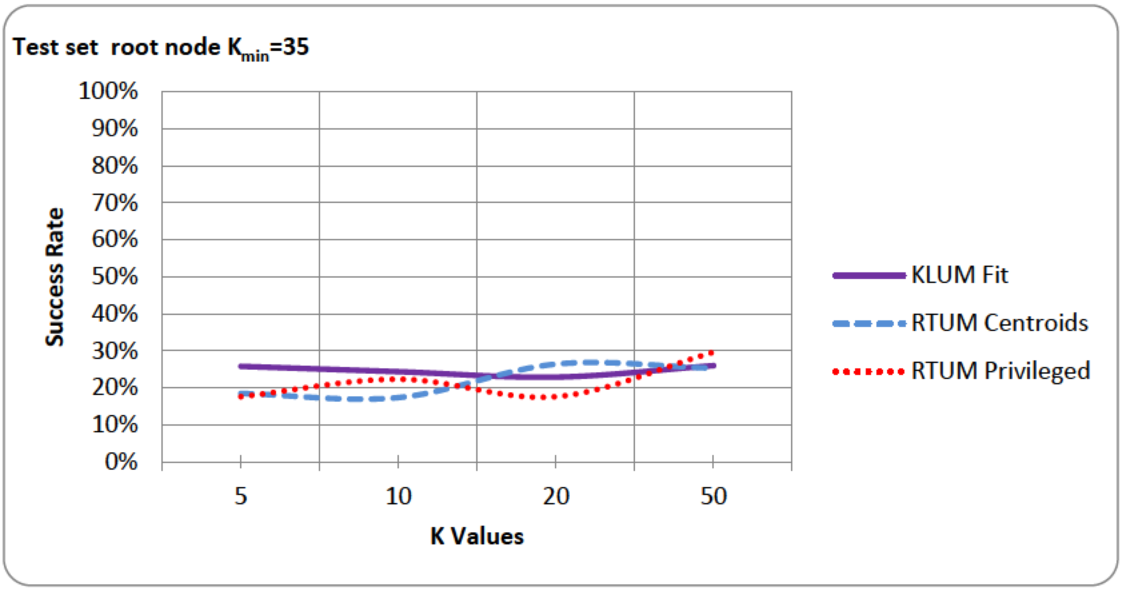
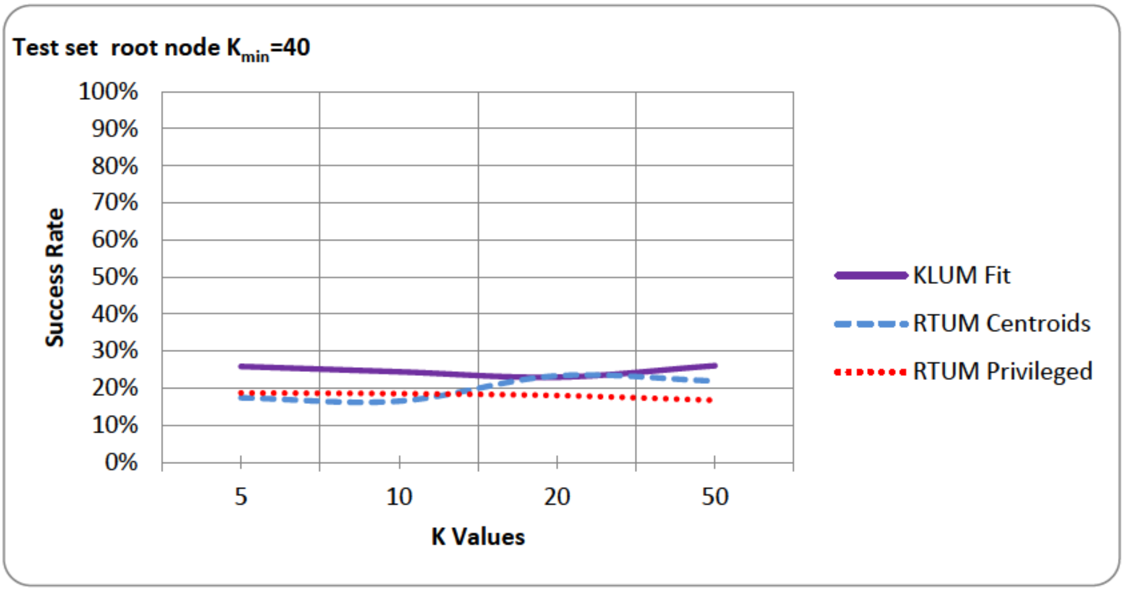


Figure 78 can be directly compared to Figure 69 where RTUM models are allowed to use their whole structure and again using the whole three performs better than using the root node only, on average the centroids model experienced a gain of 3.74% in success rate whereas the privileged mode gains a 3.44%. As in the previous experiment, the gap between both RTUM models and KLUM seems smaller when using multi-valued features and alike previous experiment all models experienced an average 20% drop in success rate due to multi-valued features. Figure 79 contains the results of the last experiment on the root node where the value of K_{min} is set to 40.

Figure 79 Root node test set multi-valued K_{min} = 40



Once again the three proposed models drop their performance due to multi-valued features, being the privileged mode the least impacted model (21.3% drop) while the centroids model is the most impacted (24% drop) with KLUM following very close (23.23% drop). After these experiments, it can be concluded that dealing with multi-valued features has an average impact of 20% drop in success rate for all models tested. When the root node experiment is compared to the equivalent whole tree experiment, both RTUM models experience a massive performance gain, leaded by the centroids model with a 19.38% gain and followed by the privileged model with a significant 11.05% improvement. Therefore, this set of experiments proves the benefits of RTUM structure and establishes a comparison between the root node of this innovative model and a classical model (KLUM). After the root-only experiment, the next experiment proposed along the evaluation involves testing RTUM model using only the leaf nodes.

5.4.5.6 Leaf node evaluation benchmark domain

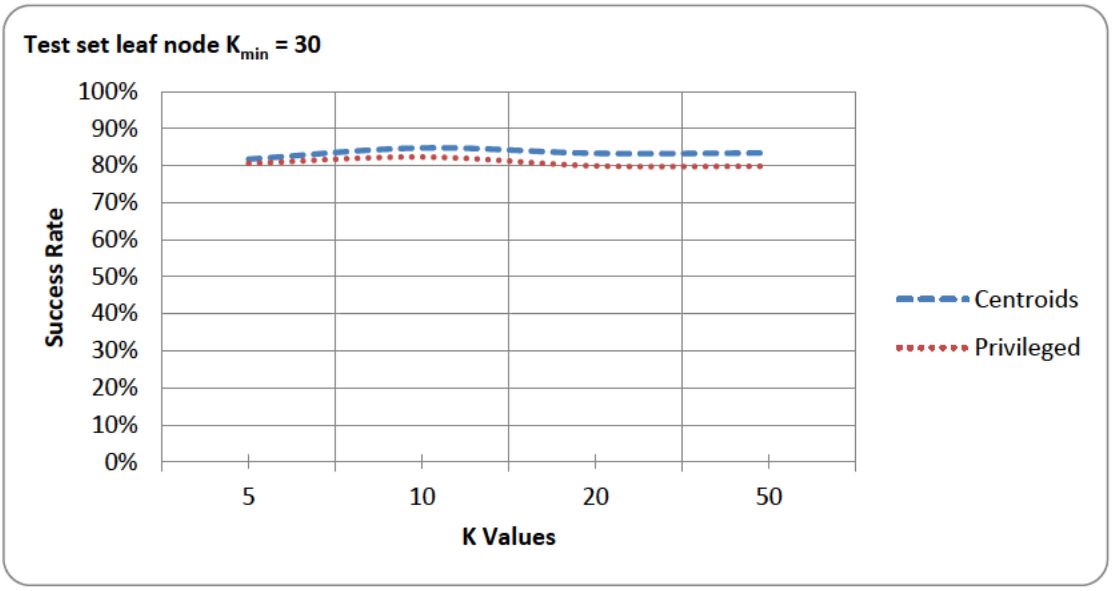
This section contains the evaluation of RTUM against the benchmark domain using only the leaf node in the R-tree structure. During this evaluation, the right leaf node is supplied to the model in order to perform the inferences. The main reason to complete this evaluation is measuring the maximum capacity of the model in terms of success rate. It has to be noted that this evaluation is performed always on the right node (this eventuality might not happen during regular evaluation) and therefore should be a good indicator of the theoretical best performance of the model. Similar to the root node experiments, RTUM models are not allowed to use their inference windows during this experimentation as they always inference on the right leaf node. Alike previous experimentation, two partition methods will be tested in order to compare their performance. Table 24 shows the parameterization of the model employed for this experimentation.

Table 24 RTUM evaluation leaf-only test set

Domain	Benchmark
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	Leaf-only
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

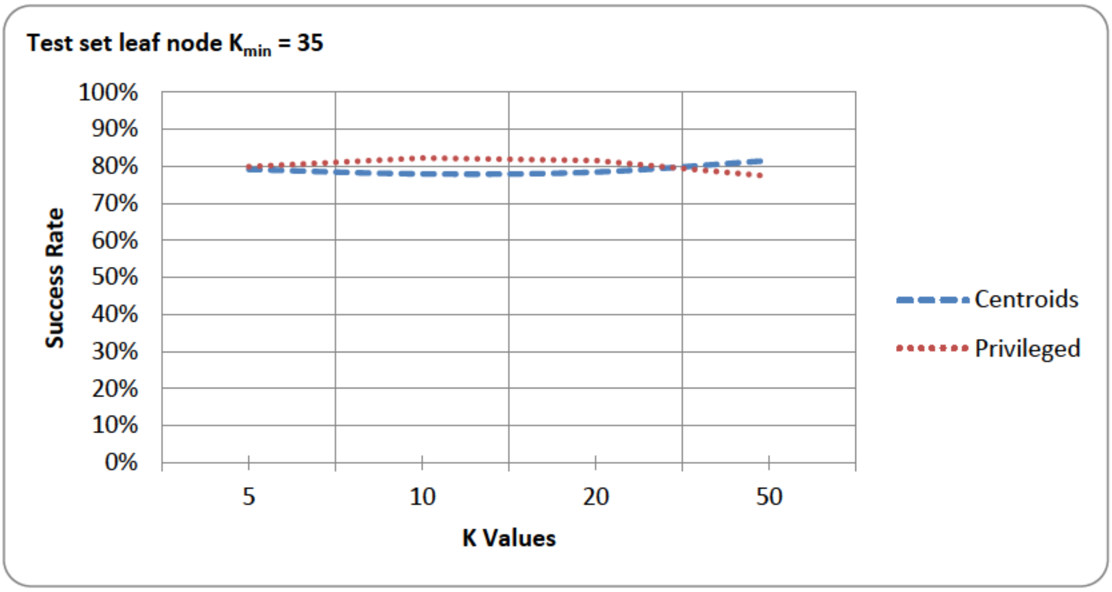
As can be seen in Table 24 RTUM model is tested against the benchmark domain using a leaf-only granularity. Figure 80 displays the results of the first experiment with K_{min} set to 30.

Figure 80 Leaf node test set K_{min} = 30



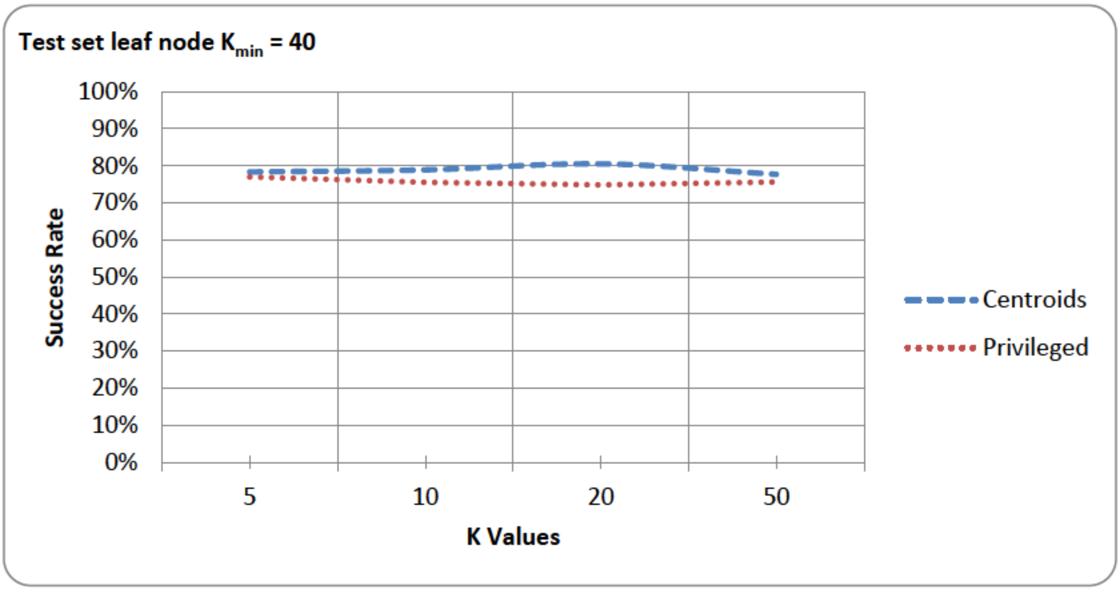
As Figure 80 shows, both models perform on par across K values being the centroids model a bit better than the analogous privileged model. It is remarkable that the centroids model reaches an impressive 85% success rate when K is set to 10 and that all its rates are over 80%. On the other hand, the privileged model also reaches success rates over 80% with a peak of 82%. When comparing Figure 80 to its analogous experiment over the test set in Figure 56, it can be seen that both models perform better when supplying the right node for the inference. Specifically, the centroids model gains a 5.68% average success rate and the privileged model a 15%. These values should be seen as the opportunity for both models in case of reaching always the right node during the inference (perfect thresholds and clusterization). Similar to previous experiments the value of K_{min} iterates. Figure 81 has the results for the next value of K_{min} (35).

Figure 81 Leaf node test set K_{min} = 35



As per the figure above, both models perform very similar and success rates are around 80% again. In terms of opportunity compared to Figure 57, both models experience a 10% gain when performing inferences on the right leaf node. As it happened for the previous experiment, both models improve when using the best node for the inferences; however, in a real-world scenario reaching this best node is not always possible due to the clusterization of the model and the parameterization. Thresholds acquired during the preliminary experiments have a huge impact on the behaviour of the inference window, where the risk of wrong choices (due to low thresholds) has to be balanced with the outstanding performance of predictions in lower levels. Besides, the fit method also affects the decision of picking the right branch of the tree and even the partition method has an impact on the clusterization that will affect the behaviour of the model. Even though, the advantages of reaching the leaf node are good enough (following the results of regular experimentation). Figure 82 shows the result for the last leaf-only experiment over the benchmark domain, as usual the value of K_{min} is now set to 40.

Figure 82 Leaf node test set K_{min} = 40



Similar to most of the experiments, the centroids partition method is better than the privileged method. In this case the improvement of the models in comparison to the equivalent experiment (Figure 58) is a respectable 5.09% for the centroids model and a 4.35% for the privileged. Results for this experiment are slightly lower than previous leaf-only experiments with lower values of K. Per the values of K, the smallest value (30) seems to outperform the rest of values (35, 40) for the leaf-only evaluation. It can be said that for this experimentation a loosened K_{min} benefits the performance of the model assuming that the right leaf node is reached. The leaf node evaluation follows with proper experiments over the social network domain.

5.4.5.7 Leaf node evaluation social network domain

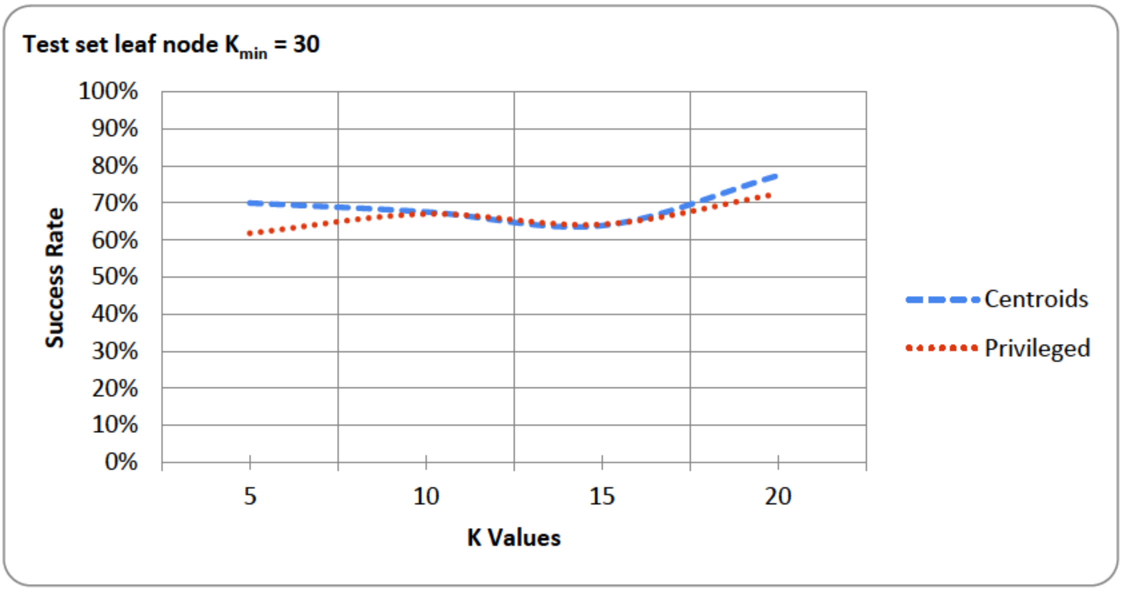
This section contains the evaluation of RTUM against the social network domain using only the leaf node in the R-tree structure. Alike previous experimentation two partition methods will be tested in order to compare their performance, also similar to previous tests over this domain two sets of experiments are proposed depending on the nature of the features to evaluate (single-valued and multi-valued). Table 25 shows the parameterization of the model employed for this experimentation.

Table 25 RTUM evaluation leaf-only test set

Domain	Social Network
K	5, 10, 20, 50
K_{min}	30, 35, 40
Granularity	Leaf-only
Inference Method	Fit
Partition Method	Centroids vs Privileged
Fit Method	Adapted K-Neighbours
Prune Method	Rows
Update Interval	Five insertions
Corpus	Test Set

According to, Table 25 RTUM model is tested over the social network domain with the particularity of capping its granularity to leaf node only. Figure 83 displays the results of the first experiment with K_{min} set to 30 and the models deals first with single-valued features.

Figure 83 Leaf node test set single-valued K_{min} = 30



Per Figure 83 it can be stated that the most popular partition method (centroids) performs slightly better than the privileged way of partition. This effect increases on both ends of the chart where the gap seems to be bigger. When comparing Figure 83 with the analogous experiment using the same K_{min}, same data sample and single-valued features (Figure 65) the centroids model improves its success rate by an appreciable 11.05% while the privileged model goes even further with a 12.16% improvement. As in previous sets of experiments, the next proposed test increases the value of K_{min} to 35.

Figure 84 Leaf node test set single-valued K_{min} = 35

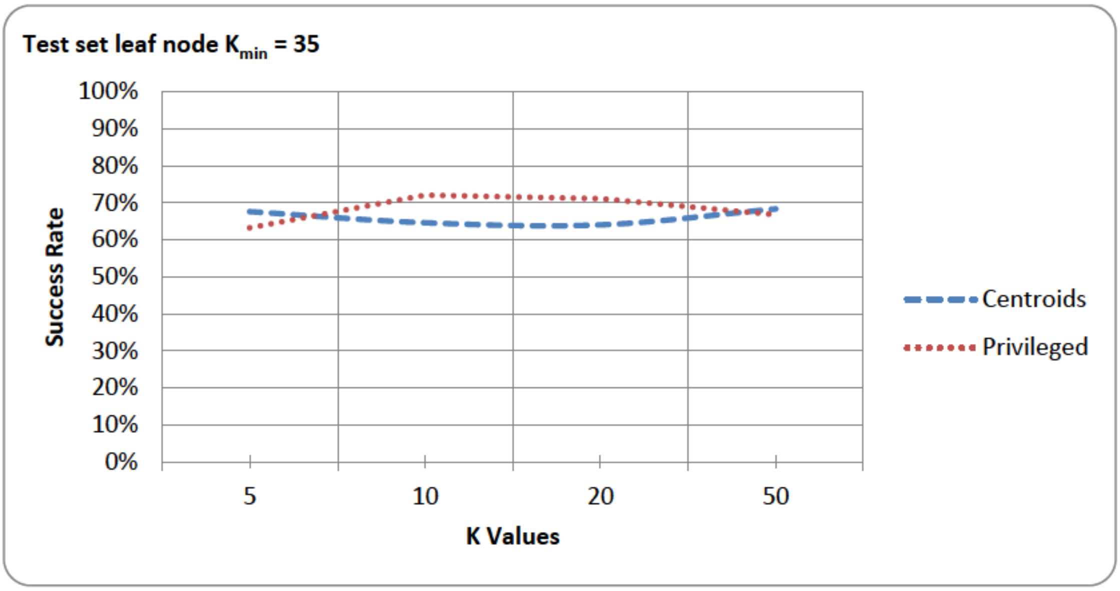
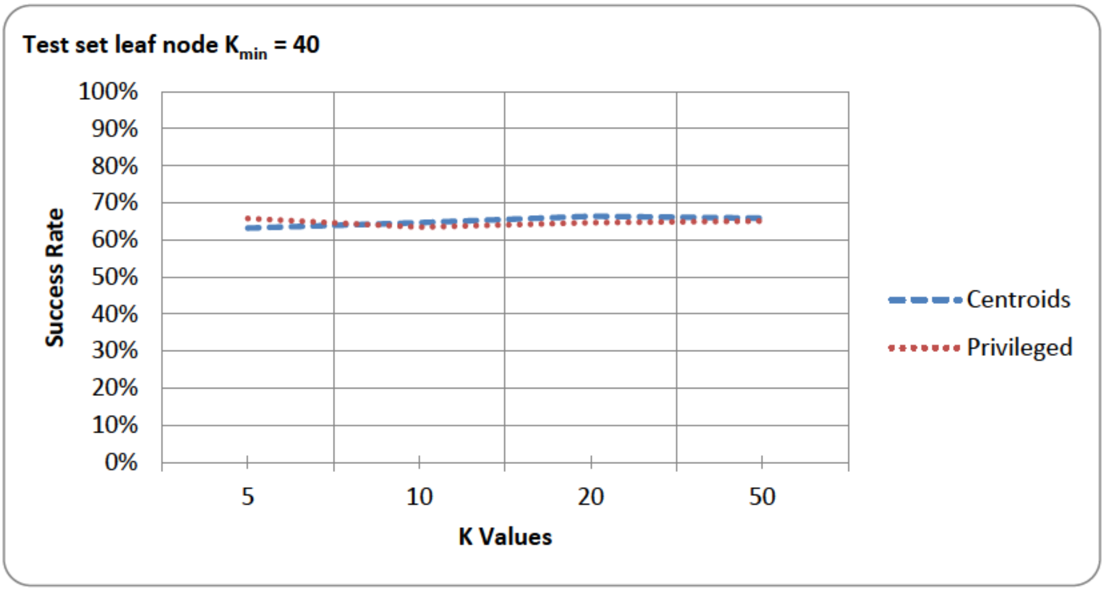


Figure 84 shows how the centroids partition method is better for both ends while the privileged method in this case seems to reach a significantly better success rate. As in previous experiments, providing the right leaf node to the models seems to boost the success rate, in this particular case results can be easily compared to Figure 66 and it can be seen as both models experience a success rate gain for each data point in the charts. The centroids mechanism has a 14.29% potential to gain whereas the privileged method can increase a 19.58%. Figure 85 has the result for the last of this series of experiments where the value of K_{min} is set to 40.

Figure 85 Leaf node test set single-valued K_{min} = 40



As can be seen in the figure above, the differences between both partition models are fairly small. The results are also almost flat for different values of K with success rates around 65% when using the right leaf node. The analogous experiment depicted in Figure 67 show poorer results for both partition methods that experience an average improvement of 13.65% for the centroids method and 17.80% for the privileged model. These results are in line with the latest experiments where the privileged model gains a bit more than the centroids sibling. That extra gain is due to inferior performance of the model (especially when using the whole R-Tree structure as in the case of the image above). Though both models perform very similar and the slight differences might be due to many different factors, the fact that the leaves experiments are similar suggests that either the thresholds or the clusterization (due to partition) is impacting the performance of this model. In order to be consistent with previous experiments, three more experimentations are done where the models have to deal with multi-valued features. Figure 86 has the results for the first proposed experiment starting whit the smallest value of K_{min} (30).

Figure 86 Leaf node test set multi-valued K_{min} = 30

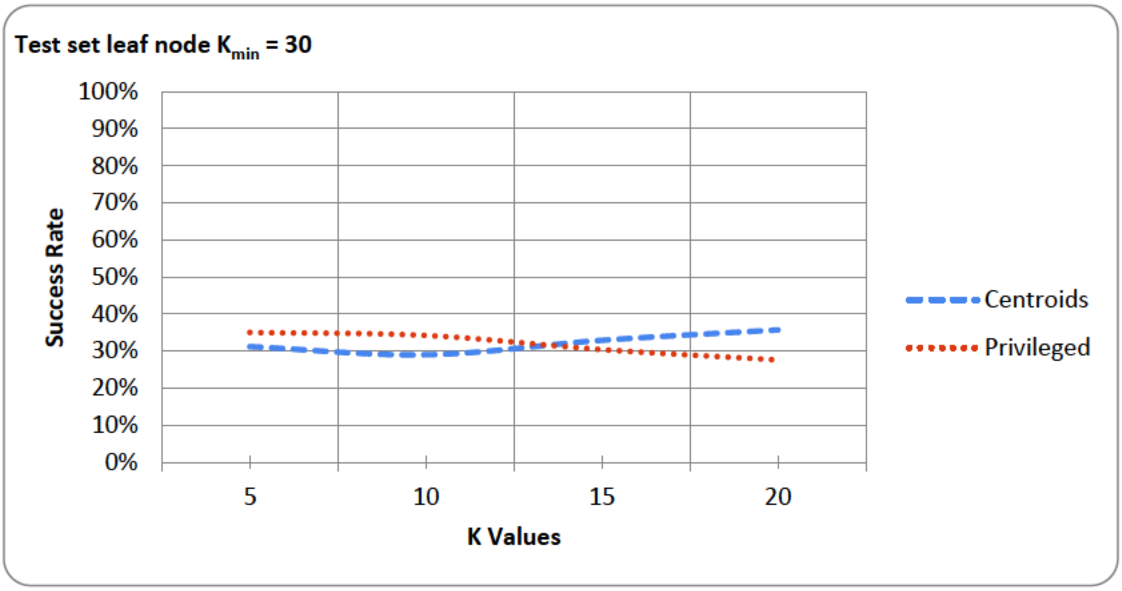
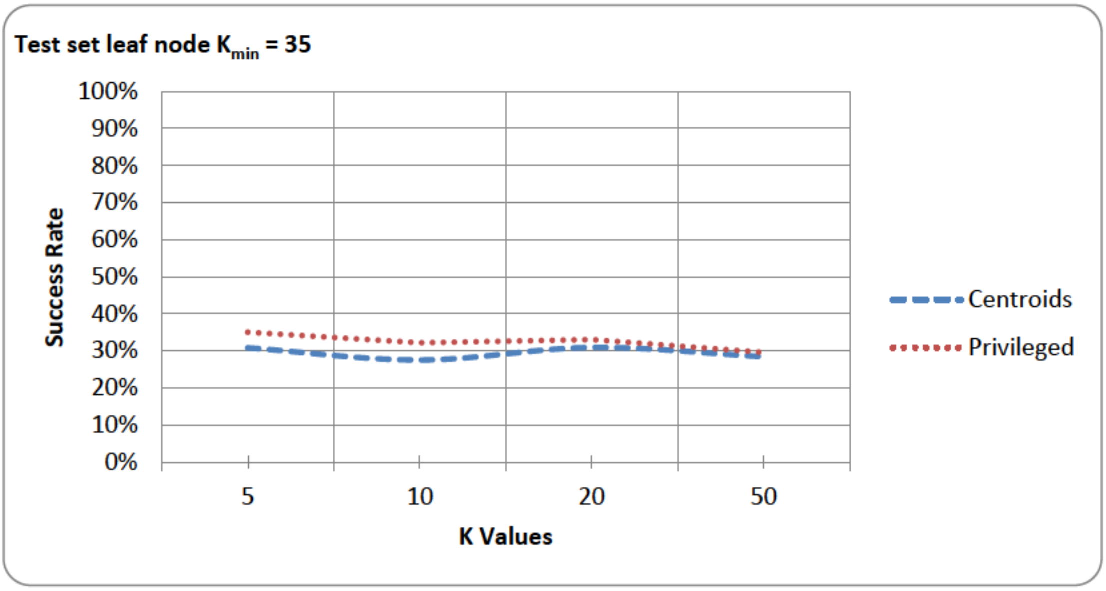


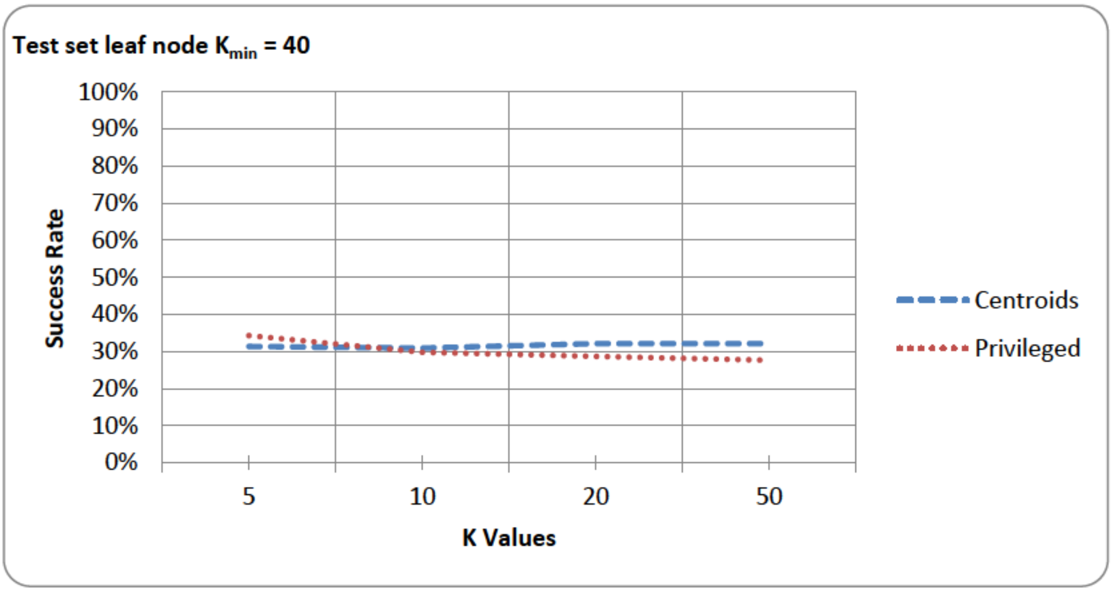
Figure 86 shows how both models perform similar being the centroids better for higher values of K while the privileged yields better success rates for lower values of K. Even when both models perform similar in terms of number of data points, the differences between both models are not insignificant being the privileged partition model a 3.8% better for the lowest value of K and in contrast, the centroids goes a head a significant 8% when K is set to 30. In comparison with the equivalent experiment depicted in Figure 68 the average success rates for all data points are very similar being the centroids partition method only 0.01% better when using the right leaf and the privileged method a 0.07% worse when using the right leaf node. These results are different to those collected in previous experiments where the models gain success rate when using the right leaf node. When analysing these close results, two plausible explanations have been found. First one is that the analogous experiment using the whole tree did quite well in terms of reaching the right leaf node. The second explanation is related to the capacity of the model to hit a prediction even when it is not using the leaves level of the tree, and furthermore if it is inferring in a suboptimal branch of the tree. Finally, after analysing several individual cases, it can be stated that the real cause is not one of those standalone, but a combination of both of them. Figure 87 has the results of the next experiment with multi-valued features.

Figure 87 Leaf node test set multi-valued K_{min} = 35



According to Figure 87, both models perform very similar and stable across different K values. Results are comparable to previous experiment with K set to 30. The same experiment but using the whole tree for the inferences can be found on Figure 69, results are better when performing the inferences over the right leaf node allowing the centroids model to gain a 1.79% average success rate whereas the privileged model has a potential improvement of 7.78% when using the whole tree. As in the majority of previous experiments over the leaves level, both models experience a significant improvement in success rates. The last proposed experiment over the leaves is depicted in Figure 88 and tweaks the value of K to its maximum of 40.

Figure 88 Leaf node test set multi-valued K_{min} = 40



Along with Figure 88 both partition methods provide success rates in the thirtyish and therefore similar values to previous experiment on the leaves level and multi-valued features. Though the smallest value ok K provides better peak performance, the three experiments deliver similar rates. In terms of comparison with the equivalent experiment over the test set and using the inference window the centroids partition method performs a surprising 7.53% worse over the leaf node and the privileged method in contrast is a 1.07% better. As previously explained, inferring on the leaf node not always provides the right answer and also inferring on as suboptimal node can also provide a good result. All in all, after all the experiments on the leaves node, it can be concluded that reaching the leaf node and performing inferences over the leaf node is beneficial and boost the performance of the models.

5.4.5.8 RTUM set against classical modelling

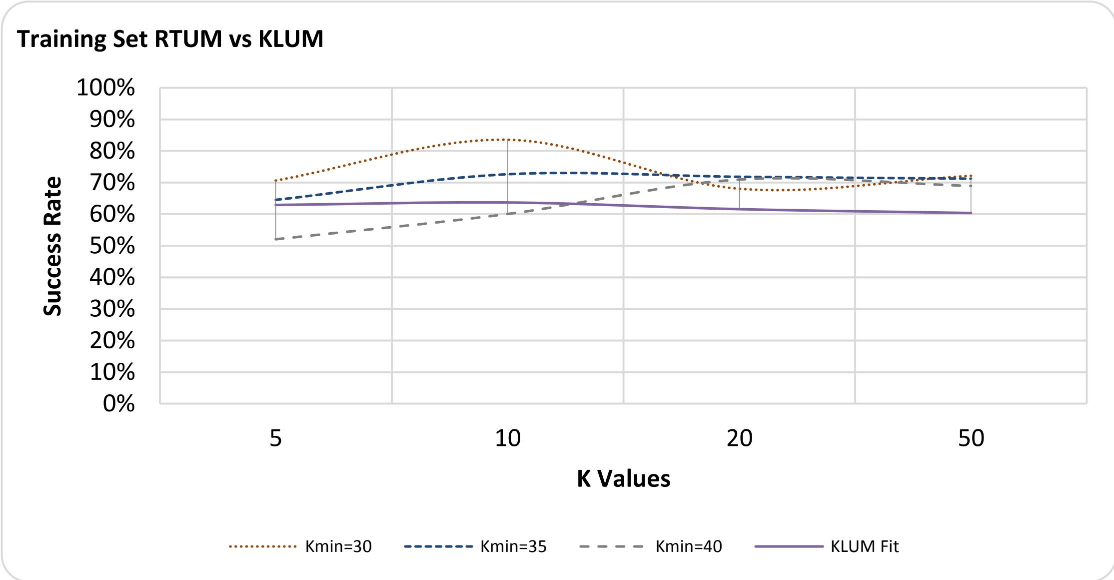
Along this section a comparison between RTUM and a classical model (KLUM) can be found. This section aims to provide a better understanding of the improvement of RTUM versus a classical and equivalent approach. Even though, just the best parameterization of RTUM is needed for the comparison for completeness reasons different parameterizations of RTUM are included on this comparison. The comparison is developed over the benchmark and social network domain and it will iterate the values of K and K_{min} per the values used during the evaluation. Table 26 shows the parameterization of both models for the comparison, although some parameters of RTUM iterate during the comparison, the best parameterization of KLUM is chosen for this comparison.

Table 26 RTUM versus KLUM

Model	RTUM	KLUM
Domain	Benchmark and Social Network	Benchmark and Social Network
K	5, 10, 20, 50	5, 10, 20, 50
K_{min}	30, 35, 40	-
Granularity	#Iteration	-
Inference Method	Fit	Fit
Partition Method	Centroids	-
Fit Method	Adapted K-Neighbours	Adapted K-Neighbours
Prune Method	Rows	Rows
Update Interval	Five insertions	Five insertions
Corpus	Training Set and Test Set	Training Set and Test Set

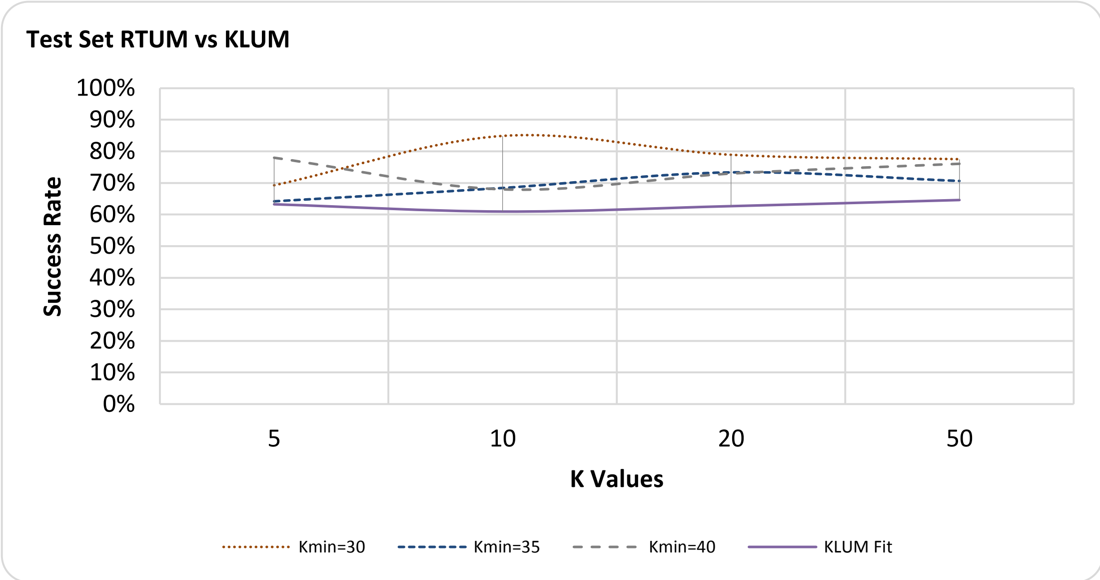
This comparison starts with the benchmark domain and training set. Figure 89 has the results for the first experiment of the comparison over the training set data sample.

Figure 89 Training set benchmark comparison



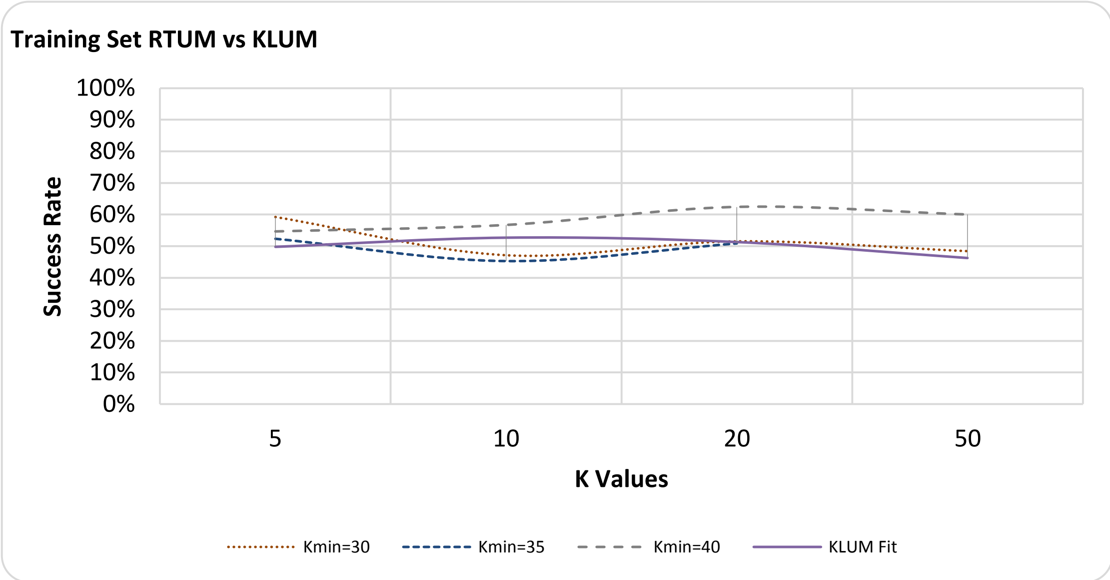
As can be seen in Figure 89, RTUM models clearly outperform KLUM for most of the data points. It has to be pointed out that different K values and K_{min} values are included in the comparison to provide a more relevant feedback of the gap between both models. In a real world application, the best parameterization of RTUM should be picked and in that case, the different in success rate if a whopping 19.85% when RTUM is parameterized as (K = 10, K_{min} = 30). The 19.85% gap is measured against the best KLUM performance which is also (K = 10). This means that using RTUM rather than KLUM provides a 20% increase in success rate for this experiment. It is also noticeable that two RTUM models outperform KLUM for each K value whereas the third one seems to struggle for the smallest K configurations but finally outpaces KLUM. To confirm these results Figure 90 shows the equivalent experiment on the test set of the benchmark domain.

Figure 90 Test set benchmark comparison



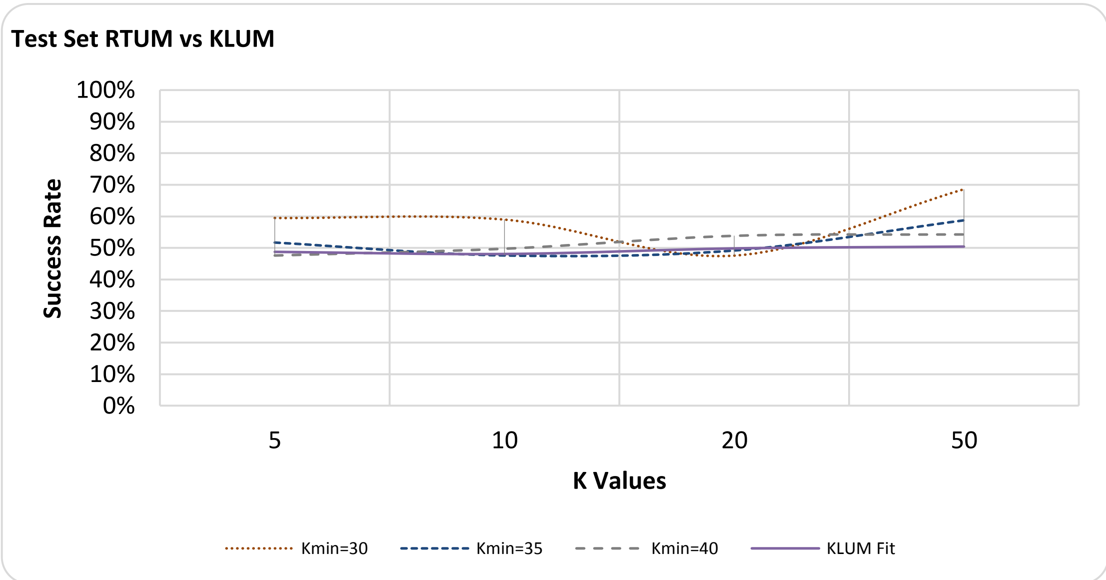
Similar to previous experiment, Figure 90 shows how RTUM models beat KLUM thus confirming the previous improvements in success rate. As it happened on the training set, the best RTUM mode is configured with K_{min} set to 30 and K set to 10. On the other hand, the best result of KLUM is produced when K is set to 30. The difference between these two parameterizations is a massive 20.32% which is similar to the improvement experienced over the training set. From these two experiments, it can be stated that RTUM models perform regularly better than KLUM, and that best parameterization for RTUM is ($K = 10$, $K_{min} = 30$) for both training and test set whereas KLUM best parameterization is $K = 10$ for training set and $K = 50$ for the test set. In order to be consistent with previous experiments, the same experimentation over the social network domain is provided next. First two charts with single-valued features and then two more with multi-valued features. Figure 91 contains the results for the training set and dealing with single-valued features.

Figure 91 Training set single-valued comparison



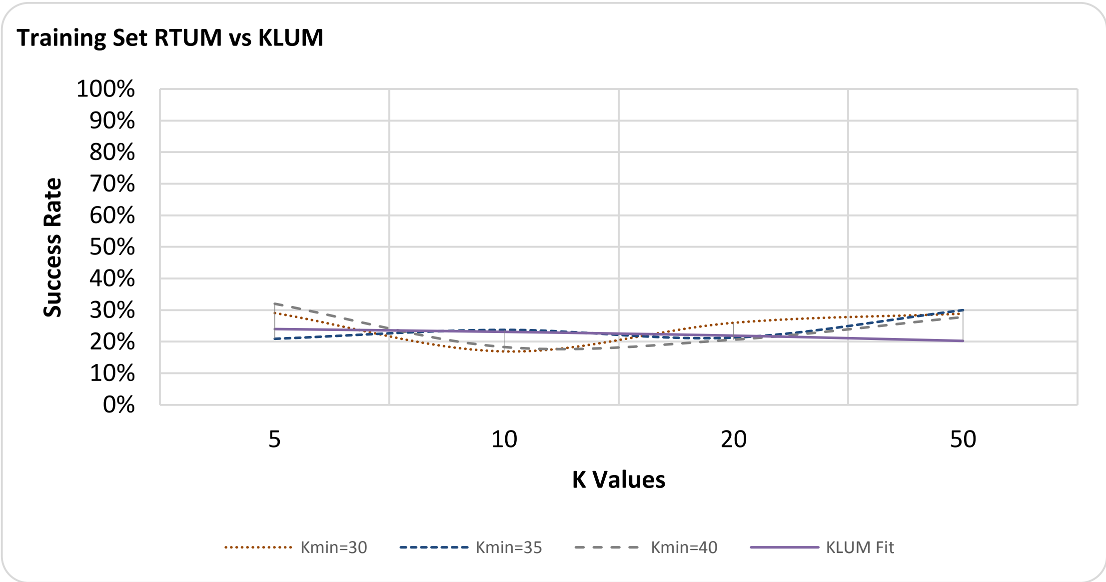
Interestingly, the gap between KLUM and RTUM seems to have decreased in the above figure. However, the best RTUM model (K = 20, K_{min} = 40) and the best KLUM (K = 10) still have an impressive gap of 9.7%. In fact, this model is regularly better than the KLUM model and therefore it can be stated that RTUM improves the success rate also on the social network domain. Figure 92 contains the equivalent experiment over the test set.

Figure 92 Test set single-valued comparison



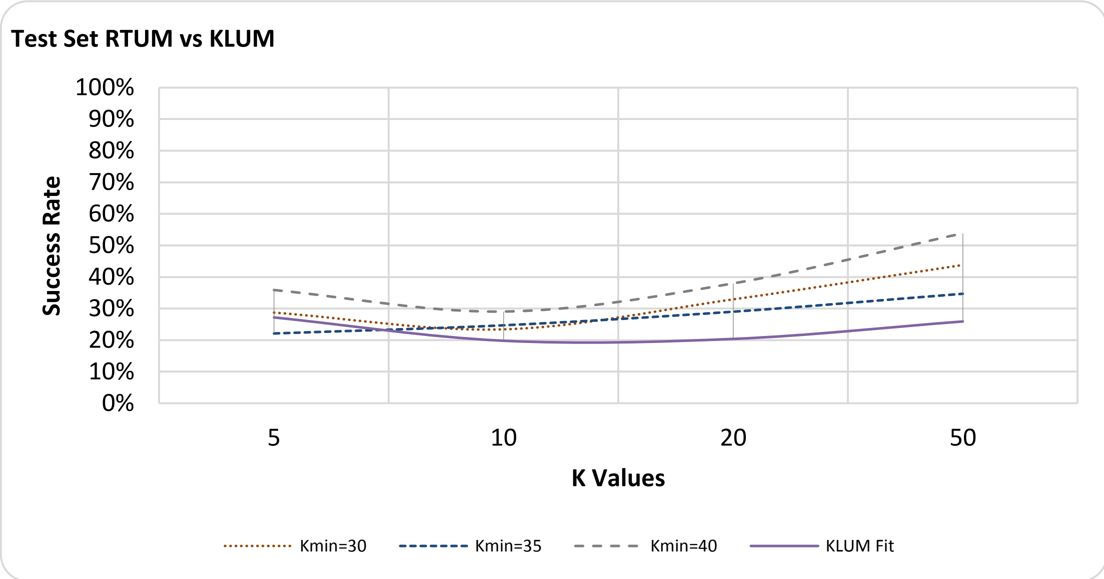
As shown in Figure 92, RTUM outperforms KLUM, the smallest K_{min} (30) outpaces KLUM with a significant improvement in success rate. The best RTUM (K = 50, K_{min} = 30) outperforms the best KLUM (K = 50) by an impressive 18.13%. Therefore, it can be said that the gap between both models is consistent as long as the best parameterization of RTUM is picked. In addition, picking a RTUM model with (K = 5, K_{min} = 30) for both the training set and the test set yields an average improvement in success rate of almost 10% over the best KLUM performance for these experiments. The following experiment depicted in Figure 93 deals with multi-valued features rather than the previous single-valued.

Figure 93 Training set multi-valued comparison



Once again, highest success rates are delivered by RTUM models reaching a peak success rate of 32.03% while the best success rate yielded by KLUM is 24% thus keeping an 8% gap even when dealing with multi-valued features. It is also remarkable that for most data points RTUM models outperform KLUM except from K = 10 where KLUM seems very similar to the best RTUM. It is also noticeable that the success rate when K = 10 is a 23% that is far from the 32% obtained using RTUM under the same circumstances. Last but not least, Figure 94 has the results for the last experiment of the comparison where both models deal with multi-valued features and the test set.

Figure 94 Test set multi-valued comparison



The last experiment shows how RTUM models perform massively better than KLUM, especially the best RTUM (K = 50, K_{min} = 50) reaches a 53.8% success rate while the best KLUM can only deliver a 27.2% for the lowest value of K. Hence, RTUM can reach a 26% improvement when compared to KLUM. In terms of choosing the best RTUM for the latest two experiments, the best parameterization is (K = 50, K_{min} = 40) as this model performs the best over the test set but also delivers a good success rate over the training set.

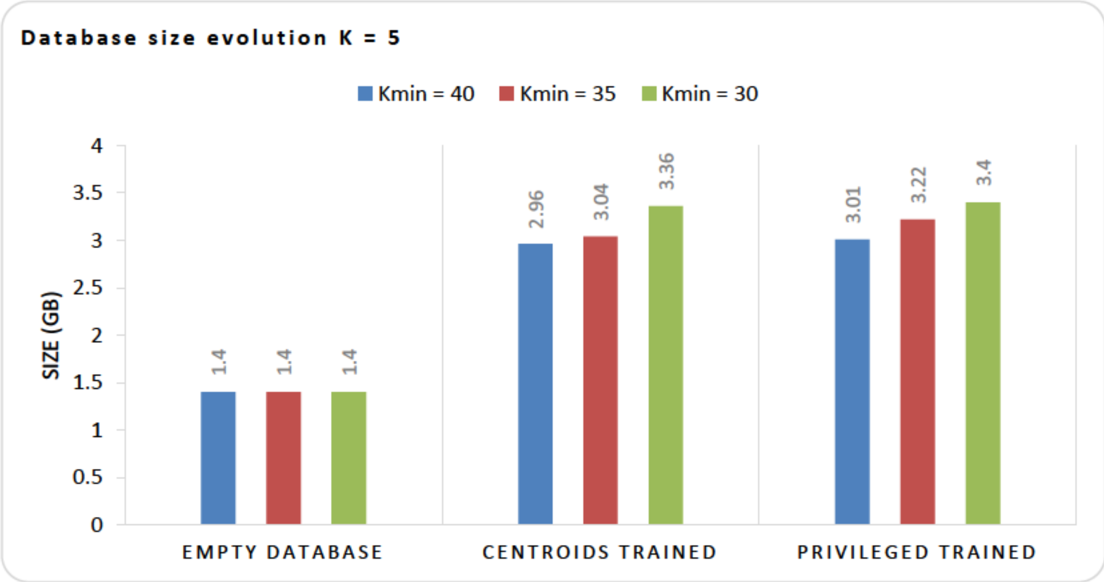
After this comparison, it can be stated that irrespectively of the domain and the nature of the features or data sample used for the evaluation, RTUM always provides better results than the classical approach (KLUM).

The next set of experiments aims to provide a better understanding on how K and K_{min} parameters impact the disk sizing when applying the model.

5.4.5.9 Database sizes

The following set of charts shows the different sizes of the databases for RTUM models before the training phase and after the training phase. Thus, iterating the values of K and K_{min} parameters it can be understood how each parameter impacts the size of the database instance. Figure 95 shows the size figures when K is set to 5 per the same hardware specifications employed during the evaluation.

Figure 95 Database size K = 5



As it can be seen in the previous figure, each centroids trained database is slightly smaller than the analogous privileged database. It is also noticeable how the sizes grow up as the values of K_{min} decrease. The size increment as the values of K_{min} decrease makes sense because the lower the value of K_{min} the bigger the unbalance is when distributing users after a partition. This unbalance means that creating a new node is more likely than it is for lower values of K_{min}. Therefore, more nodes due to more flexible distribution means bigger databases instances. Figure 96 shows analogous sizes for the same model but the value of K has been set to 10.

Figure 96 Database size K = 10

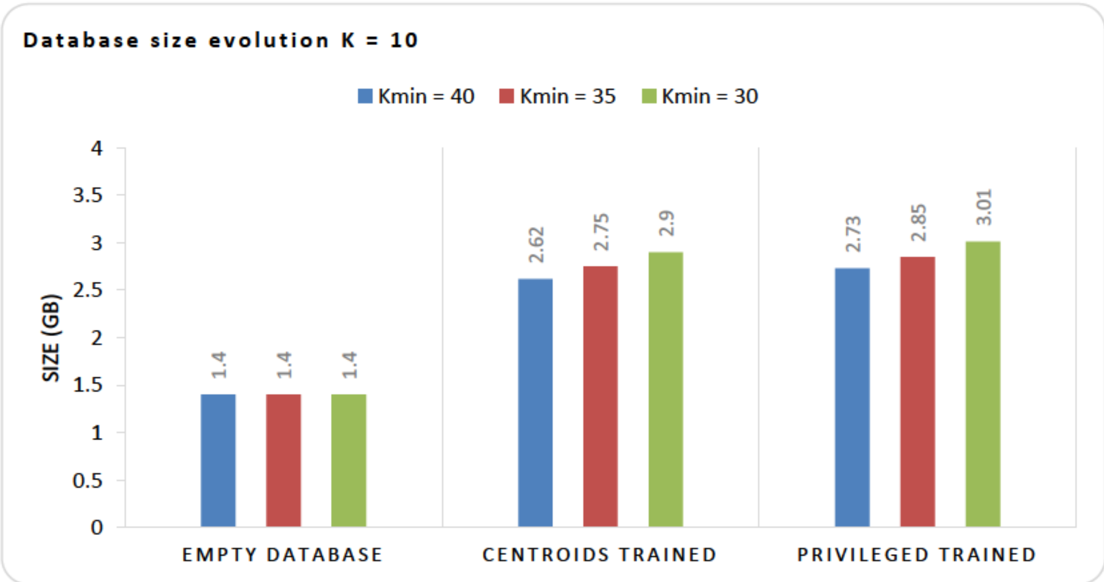
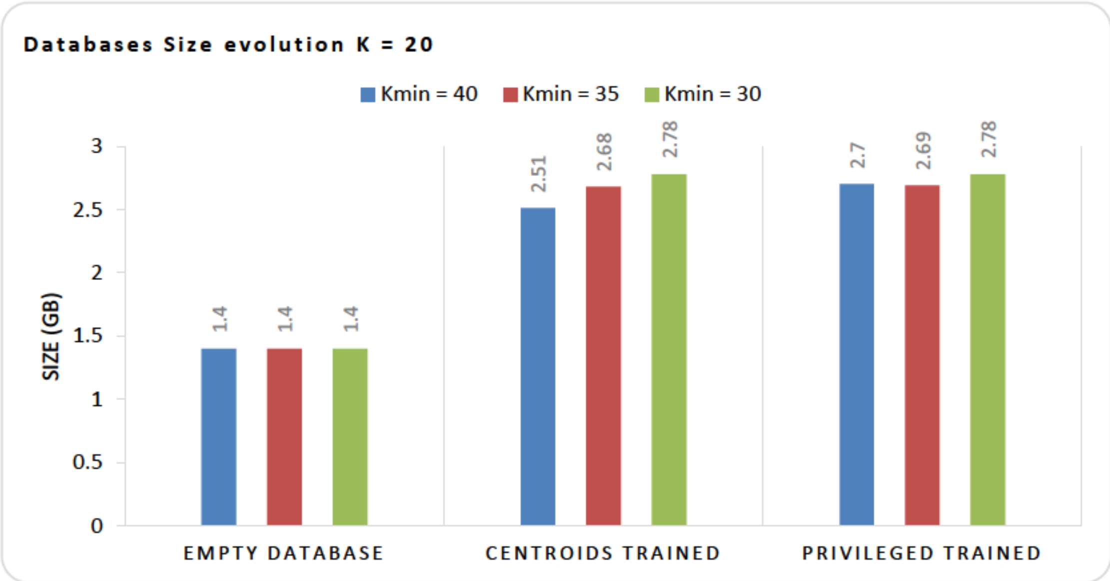


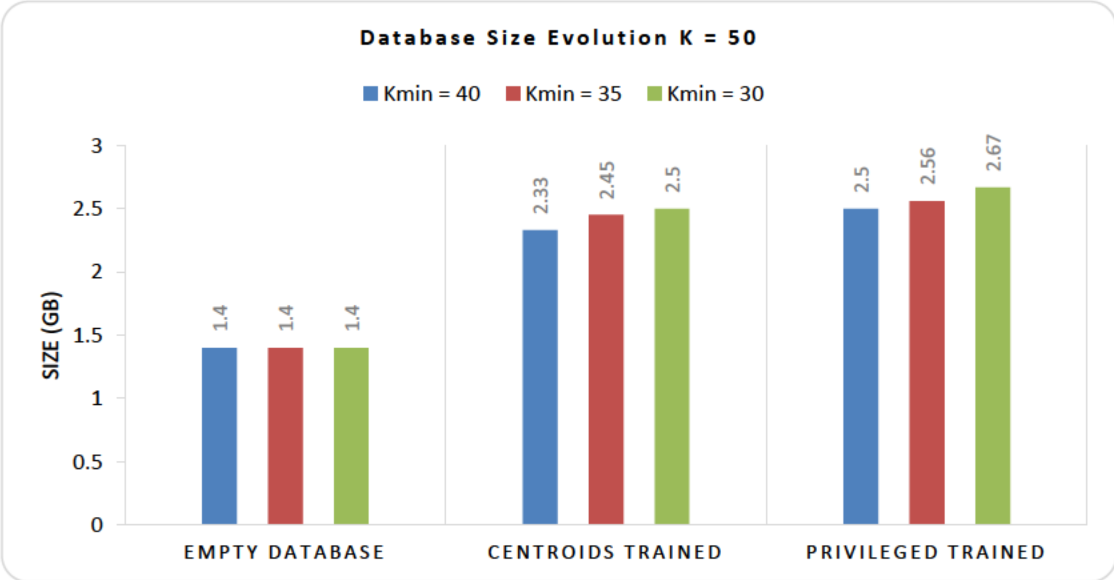
Figure 96 shows a similar behaviour regarding the values of K_{min} but interestingly when comparing one to one each trained database for K = 10, the size is smaller than the analogous database when K = 5. Again the effect of having a smaller K generally implies more nodes in the database and apparently the number of nodes can be considered crucial to determine the size of the database. The next figure shows similar databases sizes but the value of K has been increased to 20.

Figure 97 Database size k = 20



Not surprisingly, Figure 97 shows the same pattern regarding values of K_{min} and again, all databases are slightly smaller as the value of K goes up and eventually the number of nodes decrease. Finally, Figure 98 contains similar databases sizes when K is set to 50.

Figure 98 Database size K = 50



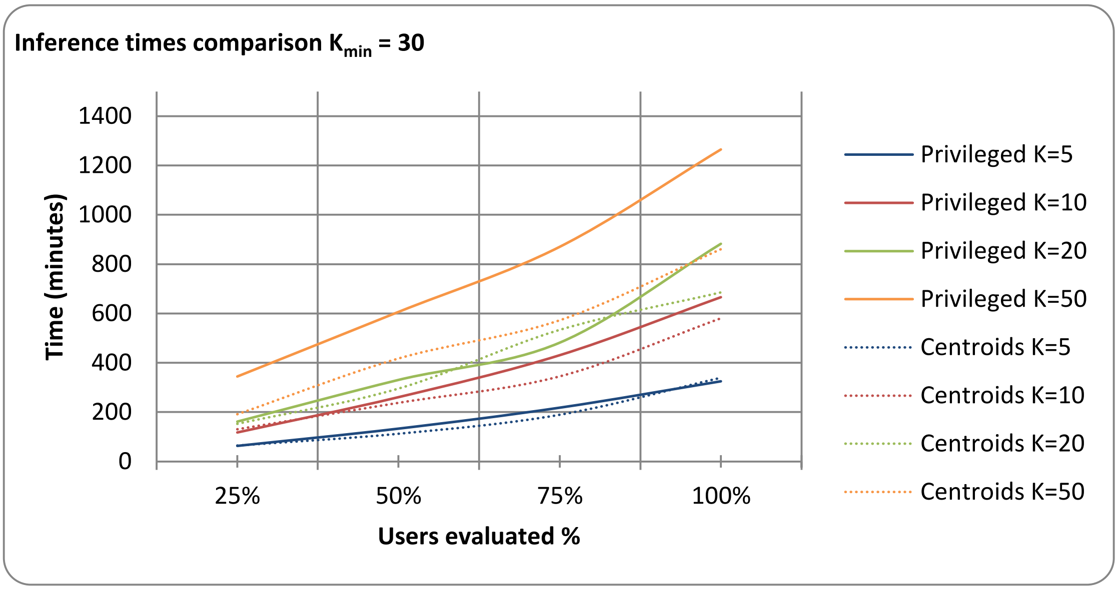
As can be seen in the figure above, the sizes of the databases instances when K is set to 50 are the smallest of all the experiments. Thus, it can be concluded that the bigger the value of K and the bigger the value of K_{min}, the smaller the database size. As previously explained bigger values of K mean larger node populations and bigger values of K_{min} mean less flexibility when distributing user groups between two nodes.

In conclusion, bigger values of K and K_{min} lead to fewer nodes in the R-Tree structure which implies smaller databases. Once the impact of the model parameterization on the disk size is proved, the nest series of experiments show how those parameters affect the inference times.

5.4.5.10 Inference times

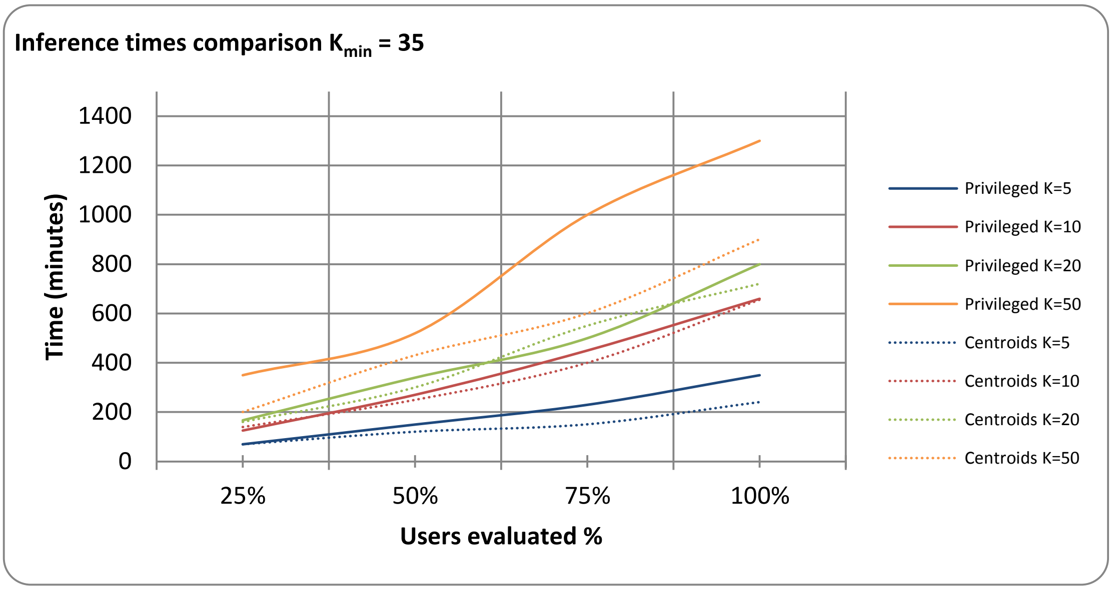
The following set of experiments aims to provide a sense of how the inference response times vary when tweaking the values of K and K_{min} for RTUM. It is important to notice that all experiments have been done using exactly the same hardware, software and with similar server condition across experiments to make the comparison relevant. Thus, the next three charts provide inference response times for RTUM when tested against the benchmark domain test set. Figure 99 shows the inference response times when K_{min} is set to 30.

Figure 99 Inference times K_{min} = 30



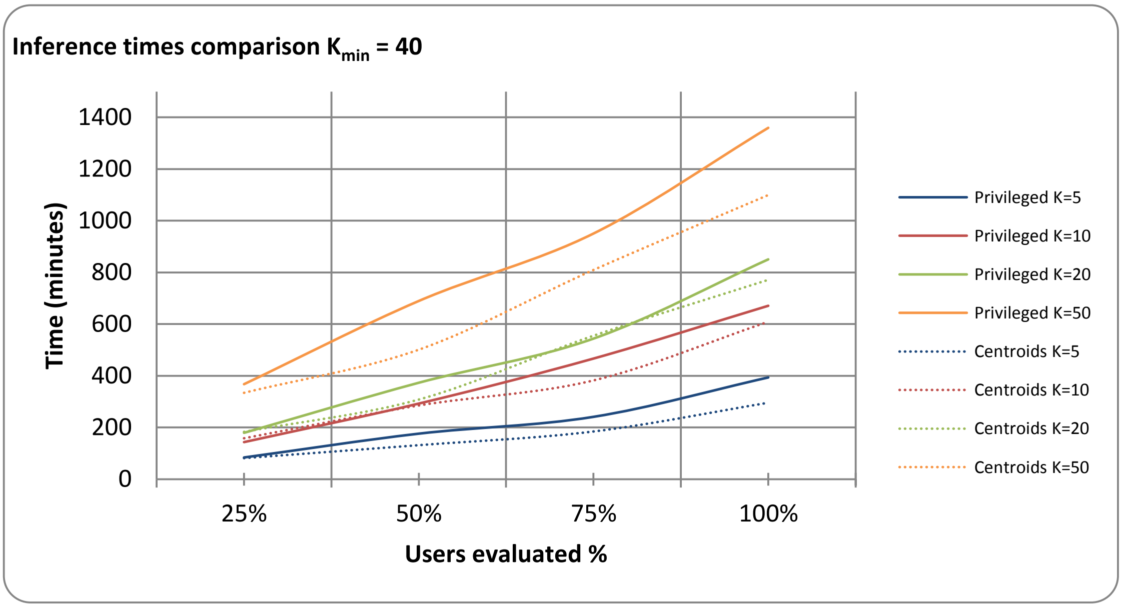
According to the figure above, the slowest model to complete the evaluation has a value of K of 50 and the top performers have the smallest value of K. As K determines the maximum population of each node, it can be stated that bigger nodes lead to slower inferences. Figure 100 contains a similar chart but RTUM parameterization has changed and K_{min} is now set to 35. Interestingly, the response times to complete the whole evaluation (100%) are on average 3% quicker than the analogous model with K_{min} = 30. While K_{min} set to 30 means that the distribution of elements can be thirty percent in one node and seventy percent in the other one, a value of 35 for K_{min} tightens the unbalance allowing a minimum occupancy of 35%. The bigger unbalance is found when K is set to 30, making new partitions slightly more likely and therefore having a certain impact in the number of nodes and partitions. This explains why the inference response times are a bit slower.

Figure 100 Inference times K_{min} = 35



Once again the RTUM model based on centroids and K set to 50 is the slowest model whereas the same model based on privileged features is the follower. Then both models with K fixed to 20 and it can be said that as K values decrease the performance increases. From these two experiments it can also be determined that those models where the partition method is set to centroids are slower than equivalent models based on privileged features. Finally, Figure 101 shows the inference response times for the biggest value of K_{min}. Once again as the value of K_{min} increases the inference response times seem to be higher. In fact, inference response times for all the evaluation (100%) are, on average, a significant 9% slower than those experiments with K_{min} set to 35. Therefore, it can then be stated that both K and K_{min} play an important role on inference response times.

Figure 101 Inference times K_{min} = 40



According to previous figure the slowest model is once again the centroids model, with highest value of K and the best performers are those models with lower values of K. From this set of experiments it can be learnt that K and K_{min} values have an impact in the number of nodes (as seen in the databases size experiments) and the number of nodes has an influence on the inference response times. In addition, the slower response times for those models based on privileged features can be explained in terms of database size, as seen in the previous section (Section 5.4.5.9) the centroids based models are bigger in size and therefore the complexity of this clusterization explain the gap in performance. After the evaluation on the benchmark and social network domain and recalling section 5.2 of this document, the models are evaluated in four different domains. Former evaluation is been done over the benchmark domain and the social network domain, the next section provides the results of the evaluation of both RTUM and KLUM on the open domains.

5.4.5.11 Open domain evaluation

The last section of this evaluation involves testing both proposed user models on open domains. This type of domains is not the main focus of the evaluation as in real world applications the models usually deal with bounded and reduced domains. However, these results are believed to have an important research value and provide a wider perspective of the performance of the models.

These domains have two main features: one is the massive amount of multi-valued features and the other is the lack of features (information atoms) for some of the users. These issues usually have a great impact on the clusterization and therefore on the performance of the models. Two different domains are proposed for the open domain evaluation, the first one is an open real case domain. This domain is available from real data and has not been corrected or validated. Thus, it can be found that two or more terms refer to the same concept, typing errors are not corrected and abbreviations can also refer to the same concept. Because of this, the second proposed open domain is a machine-generated sample that has cross-ontology validation. Even though, this second domain corrects some of the issues of the real application domain, it is still a very complex domain due to its nature and amount of multi-valued features. Table 27 shows the parameterizations of the models to be tested.

Table 27 Open domain evaluation

Model	RTUM	KLUM
Domain	Open domains	Open domains
K	50	50
K_{min}	30	-
Granularity	#Iteration	-
Inference Method	Fit/Fit and certainty	Fit/Fit and certainty
Partition Method	Centroids	-
Fit Method	Adapted K-Neighbours	Adapted K-Neighbours
Prune Method	Rows	Rows
Update Interval	Five insertions	Five insertions
Corpus	Training Set	Training Set

According to, Table 27 both models RTUM and KLUM are tested on both proposed open domains. First part of the evaluation is done over the open domain and main results are depicted in Table 28.

Table 28 Open domain evaluation results

Model	Fit partition method	Fit & certainty partition method	Domain default value
KLUM	12%	14.7%	0.01%
RTUM	14.3%	15.02%	0.01%

According to Table 28 RTUM user model outperforms KLUM irrespectively of the partition method used. This result just confirms previous results in different domains where RTUM consistently performed better that KLUM. It is also remarkable that even when the results seem relatively low compared to previous domains the domain default value is an insignificant 0.01% compared to 16% for the benchmark domain and a 0.45% for the social network domain. Thus, results on this domain are expected to be lower than those obtained on previous domains. Also in favour of these results it has to be said that the open domain has an impressive 81.8% of multi-valued features plus multiple non-validated values and some very difficult to classify users due to lack of information about them. Once again, evaluating the models on this typology of domains has a strong relevance for the academy and research purposes but lacks of impact on the industry or real world applications due to the nature of the domain. Taking all this in mind, the results obtained are good results that show the complexity of the domain and the great impact of multi-valued features and non-standardized domains. To follow up with the evaluation, Table 29 shows the results obtained for both models on the artificial domain.

Table 29 Artificial domain evaluation results

Model	Fit partition method	Fit & certainty partition method	Domain default value
KLUM	18.68%	18.72%	0.1%
RTUM	20.31%	21.89%	0.1%

The artificial domain is an open domain with some cleaning processes supported by an ontology that solves the problems of synonymy and syntactical errors. However, the average feature cardinality and the amount of multi-valued features is still very high and has an impact on the performance. As displayed on Table 29 results on this domain are higher than those obtained on the open domain as it is the domain default value that is ten times higher for this domain. Alike the previous domain RTUM provides the best success rates for the domain thus confirming its superiority across four different domains.

5.4.5.12 Experiments Analysis and discussion

As a first and global result of the evaluation, it can be stated that RTUM model outperforms KLUM across the four proposed domains. Being the latter a representative and general implementation of the classic approaches for statistical user modelling, it can be followed that the RTUM approach offers some benefits that can be advantaged either by applying the model as a whole or by taking some isolated aspects of it into another already existent models.

Furthermore, the diversity of the datasets applied through the evaluation demonstrates that the model proposed is versatile enough, and adaptable to different needs and domains. The performance of current implementation of the model is not only comparable to other similar systems, but shows state of the art results in many cases, some of which have already been published in international widespread and well indexed journals.

This discussion of results aims to extend the understanding of the evaluation, state the implications, and summarize strengths and weaknesses. To ease the reading of this section, the conclusions have been distributed into six main issues, briefly exposed next.

- *Testing alternative methods*

Going deeper in detail, it should be pointed out that the centroids partition method outpaced that other one based on privileged features for the majority of experiments along the evaluation of the RTUM. However, to the extent of the evaluation carried out, it is not possible to assure there is no specific scenario where the latter partition method performs better than others. Indeed, such method success is strongly linked to the correlation between the chosen privileged features and the afterwards queried features. In general scenarios, it is hard to find a feature (or small sequence of features) showing good correlation to all queried features, while the joint of many other features lead to better characterization. Nevertheless, it is interesting to take it into account for those specific cases. Besides, the privileged partition method not only provides good enough success rates (excellent in some cases), but also an outstanding clusterization that is especially useful in some domains. A good example of this is found in the social network domains, where apart from providing accurate predictions the RTUM with that partition method (based on privileged features) can help in detecting communities within the social network. Thus, the added value of this partition method can justify its election in some specific cases.

In regard of inference methods, two main approaches have been tested and it can be concluded that the so called *maximum fit method* performs better across different domains (Benchmark and Social) but the fit and certainty is slightly better on open domains where the features to infer are defined on excessively wide ranges. However, it is fair to say that the maximum fit method came always close on the open domain experiment with an average loss of 1.71% on the open domain and a negligible 0.8% on the artificial domain.

- *Results regarding the parameterization of the model*

In respect to K and K_{min} parameters, it is hard to settle a universal recommendation, as the results vary across domains and experiments. However, having a deeper look into the results, it can be said that regarding the benchmark domain and to the extent of this evaluation, the best K_{min} value found was 30, as it delivers an average success rate of 73.55% on the training set across different values of K and a 77.63% on the test set. These results are 6% better than those obtained when K_{min} is set to 35 and a 14% better than the same results when K_{min} is 40. When tested on the social network domain, conclusions about the best K_{min} are even more difficult to make as performance varies depending on the data sample used to evaluate (training set/test set). On one hand, and irrespectively of the nature of the features (single-valued, multi-valued), the best value on the training set is again 30 (the lowest). But on the other hand, when testing against the test set, K_{min} set to 40 (the highest) is slightly better than 30. Finally, concerning the best K value all K values perform similarly but fixing the value of K_{min} to the previously discussed maximum (30) the best K is 10 which actually delivers the peak results reaching an 83.5% on the training set and an 84.9% on the test set. The social network domain is again harder to make a decision but having set a value of 30 for K_{min} the best K value is 5 but differences are very small.

Therefore, due to its variability these values should be trained and decided in preliminary experiments for each single domain. However, it has to be observed that most of the values tested provided comparable results making it possible to migrate the model across domains and get competitive results.

Another level where the parameterization of the model has a strong impact is the size of the database supporting the model. During the evaluation, it has been stated that, as expected, the bigger the value of K and K_{min} (tighter distribution), the smaller the database holding the knowledge bases supporting the model. This result was foretold upon the following basis: the bigger the number of nodes the bigger the size of the database; smaller values of K (maximum number of individuals in a single node) mean more nodes for the same population and, finally, relaxing the value of K_{min} provides more flexibility but also increases the chances of new partitions and decreases the occupation density (hence increasing the number of nodes for the same population). Finally, it should be added that the privileged-features partition method produces bigger database size (if compared to the centroids method) due to its more complex clustering.

Model parameterization also impacts both the inference response times and the model global performance. On one hand, bigger nodes cause the performance of the model to be slower, as proved along the experimentation, being the models configured with smaller K values significantly quicker than any others. On the other hand, the bigger the database is (in number of nodes), the more imperative the need for going down through the tree structure in order to attain good enough results, also deriving in eventual waste of time. In this line, it has been stated that the privileged-features partition method draws globally worse response times.

- *Highlights on quantitative results*

Reviewing the numerical results of the RTUM evaluation, the proposal achieves state of the art success rates, reaching peaks close to 85% (Figure 53 and Figure 56) in the benchmark domain, success rates on the sixtyish in the social network domain, and 15% on open domains (nearly 22% if normalized). Yet the latter seems to be low, it should be reminded of the vast cardinality of domains, leading to hard predictions. If compared to recommender systems or to web search engines, it is like hitting best choice in first place of the results provided, and current technologies of this kind show success rates swirling around the attained by RTUM.

- *Best and worst performing cases*

In addition to general working evaluation, some test where conducted where the RTUM model worked with its inference window (the tree node focused for inference processes) forced to stay at the root node, thus obtaining the worst predictions it can provide. Through these root-node tests, RTUM showed acceptable performance, yet worse than classic approach KLUM. Indeed, it should be pointed out that the root node separated from the rest of the tree is comparable to a KLUM model (where groups of users are summarized into stereotypes, and those stereotypes used through inference processes). But the KLUM's processes are designed to work in a single node, while the RTUM's are designed for working within the whole tree.

Regarding the best performance case, a leaf node evaluation has been provided to prove the theoretic maximum capacity of the models. Per the previous experimentation, RTUM models provided impressive success rates close to 90% for the benchmark domain and close to 80% and 40% for the social network domain with single-valued and multi-valued features respectively. It is remarkable that there is not a big gap if compared to the model's general working (barely 5% improvement). For analysing this fact, it should be recalled that these results must be understood as typical behaviour of the model, since, as proved during the evaluation, inferring on a sub-optimal node or branch yield good results and often hits the right prediction, while prediction on the right node could eventually be mistaken. The gap between each maximum and the model performance is determined by different factors, but two decisions are crucial for the performance of the model, one is the parameterization as largely discussed during the evaluation and the other one is the selection of thresholds. In this document, four different thresholds have been explored with one (iteration number) performing better than the others. However, different thresholds should be explored in the future as this is critical to decide when the inference window moves down one level.

Finally, an open domain evaluation has been performed to test the model in the worst possible scenarios, yet only for the sake of completeness and with academic purposes, as this is not the main target of these models. Nevertheless, this evaluation provided very interesting results, showing that the proposed RTUM user model performs better than the classic approach (KLUM) on each of the four domains tested. Moreover, the results of KLUM on these scenarios proved to be competitive, opening a future line of work in adapting this technology to application domains where term validation is never possible. The open domain evaluation also provides a very good feedback of how multi-valued features and non-validated terms can impact the performance of both RTUM and KLUM, and potentially any other user model.

- *Comparison against classical approach*

The next consequence of the evaluation is that the comparison between RTUM and KLUM left a clear winner in terms success rate and clusterization. According to, the experimentation RTUM user model leaded the evaluation over the benchmark domain with an average 20% gap over KLUM. In addition, RTUM user model also outperformed KLUM with a noticeable 10% average improvement in success rate on the social network domain. This together with its advance structure based on an R-tree to solve the scalability problem makes the use of this model and in general any other similar approach a good decision in order to beat the performance offered by classical models. In fact, only through the root-only evaluation KLUM was able to outperform RTUM, and this was because the latter was forced to work in the RTUM style (being this one, certainly, more adequate for such working). But this fact reinforces the conclusion that the advanced (R-tree) structure, and the use on it of the inference window, is definitely more advantageous than any classical KLUM-like model.

- *Evaluation final remarks*

Summarizing, a complete evaluation has been provided with diversity of models, parameters and domains. Along the evaluation, it has been proved that the proposed RTUM model outperforms a classical user model approach in terms of success rate, clusterization and performance. Last but not least, this evaluation provides an added value as it is comparable and it is easily replicable. Indeed, there was a serious lack of unified methodologies and tools for evaluating statistical user models, and specifically, there was no benchmark for this task. The introduction of a standardized and user model oriented domain for evaluation (benchmark) makes simple to replicate the evaluation and cross-validate the results. This is an extremely important milestone, due to the lack of frameworks for user model evaluations that during many years made enormously difficult to compare results across different models, and certainly another contribution of this PhD thesis to the area.

6 Concluding Remarks

User model and therefore recommendation algorithms have become very important due to the vast amount of information available. Most companies and institutions rely on recommendation algorithms to increase their revenue, reach larger populations or target customers with the most appropriate content or product. Both technologies have their roots in multiple disciplines such as information retrieval or machine learning among others. In fact, as data becomes more and more important for companies, universities and people, user models are crucial to make decisions over large amounts of data. However, after a deep review of the literature, it was found a lack of statistical models based on experience. In addition, the existing models suffer from some major problems as the scalability problem, the cold-start problem or the new user problem. Among these problems the scalability problem is crucial due to massive amounts of data available for companies and institutions. Furthermore, the area of user modelling suffers from the lack of proper evaluation frameworks to make evaluations comparable. Thus, leaving researchers and developers with ad-hoc evaluations that cannot be compared. Therefore, per the current literature there is an overwhelming demand of statistical models to handle large datasets alongside with proper evaluation frameworks.

Current developments in the area of user modelling cover classical approaches that usually rely on content-based model. Though, those models have proven their ability to provide recommendations on medium size datasets they usually suffer from major problems in the area of user modelling as scalability, cold-start or new user problems. Additionally, those content-based models rely on current user actions and behaviours to provide recommendations. While this approach has been successfully applied during the last decades the rapid development of data science plus the large amounts of data available require new approaches able to provide better predictions and capable to handle larger datasets. Having in mind those requirements this PhD thesis focus on developing an innovative hybrid user model that combines the strengths of content based and collaborative models at the same time that tackles the problem of scalability. This PhD thesis also proposes a complete evaluation framework for researchers and developers. This framework makes evaluations comparable across different models while guaranteeing the statistical significance of the evaluations.

Along this thesis two statistical user models based on experience have been proposed (KLUM and RTUM). The first one (KLUM) is a classical approach that summarizes and remove data in order to keep its performance level within reasonable margins. The second one (RTUM) is an innovative model based on an R-Tree structure. The aim of this new model is providing accurate predictions while solving some of the biggest problems in the area of user modelling. RTUM user model thanks to its advance structure solves the problem of removing data while addressing the scalability problem. RTUM user model requires only visiting one branch of the tree to provide accurate predictions. That means the process of providing an inference can be considered depth-first as it only traverse along one branch to produce the inference. To put this into perspective, RTUM can model the world population (roughly 7 billion) with only 8 levels (given a typical parameterization $K=20$, $K_{min}=50$) and without removing information from the knowledge base. This is a massive achievement as the performance of the model won't be impacted by the size of the dataset and because no other model in the literature provides this functionality. Moreover, a user model not necessarily uses identified users but different interlocutors (one per session) so the same user produces as many registries as sessions held. Therefore, the model has different interlocutor status and different values for each session. The main downside of this approach is the huge amount of data produced. Thus, only a model able to scale can handle this type of interaction.

Both proposed models have been developed and tested with equivalent formulations to make comparisons relevant. Moreover, the models have been tested in different domains and have proved to deliver competitive success rates across domains. Thus, proving the versatility of the models and how their success rates are competitive across domains.

Regarding the proposed models, RTUM and KLUM are prepared to create their own knowledge base from scratch but also can be fed with expert knowledge thus alleviating another major problem in the area of user modelling as it is the start-up problem. While most of the classical approaches suffer from the start-up problem, both RTUM and KLUM can be fed with expert knowledge to solve the start-up problem. The expert knowledge is used to provide initial inferences and it is refined with newly acquired knowledge during interactions. When the amount of knowledge in the database grows KLUM will eliminate knowledge to keep performance within reasonable levels whereas RTUM will keep all the knowledge in its database. However, the impact on the inferences of those initial stereotypes can be reduced. If a new user that fit those stereotypes comes into the system RTUM will progressively restore those stereotypes and increase their influence on the inferences. While this seems like a normal behaviour it is a remarkable feature of the model that makes it a hybrid model and addresses the start-up problem.

RTUM user model introduces an inference window that allows the model to perform inferences with different levels of granularity. The inference window has been proved successfully along the evaluation with experiments in the leaf nodes performing stronger than any other experiment. However, focusing simultaneously on more than one node during the inference should be explored as it can boost the performance but also improve the results. To achieve focusing on two nodes at the same time a good data management supporting data sharing and parallelism is necessary. The need of a powerful database support raises the need of a broader study of databases covering physical data structures, cache policies and some other features. This will be left for further research as it is beyond the reach of this thesis.

In regards to the evaluation of both models, this thesis provides a complete evaluation where a new model based on an R-Tree (RTUM) is evaluated against four different domains plus compared to a classical approach (KLUM). The so called RTUM model proved to perform better than the classical approach on every single scenario across the four domains. RTUM reached success rates of 85% for the Benchmark domain where KLUM managed to reach a 65% which is an impressive 20% gain in success rate. Accordingly, the gap between both models on the social network domains also reaches the 14% with RTUM reaching a 68% and KLUM a 50% success rate. The evaluation provided not only compare models and success rates but also provides a broad analysis of how every parameter of the models impacts the performance plus a complete study of the databases sizes and inference times for the models. The main conclusion to the evaluation is that after a complete evaluation with a wide diversity of parameters and domains RTUM outperforms KLUM on every scenario tested.

As aforementioned, RTUM user models comes to fill an important gap in the area of user modelling where there is a lack of statistical user models based on experience. The model also addresses major problems in the area like the start-up problem or the knowledge loss. However, the main strength of RTUM is its ability to handle large datasets without decreasing the performance. If RTUM comes to fill a gap in the area of statistical user models based on experience and addresses some of the problems in this area as the scalability problem, the evaluation provided is not only complete but also replicable. The importance of replicable evaluations can be noticed after the literature review where no other evaluation framework for user modelling was found. Thus, this thesis also provides a standard evaluation framework for user modelling. The proposed evaluation framework includes data samples (training set and test set), sets of experiments a complete set of metrics to measure during the evaluation and a complete study to guarantee that each experiment is statistically significant. This framework is available for public use and can be downloaded and then used to evaluate and cross-validate results of different models. Additionally, the proposed models have been tested using the evaluation framework thus making this evaluation more relevant as it can be easily compared to any other model in the area.

However, the main downside of RTUM user model is the amount of redundant knowledge that stores in its knowledge base. As explained before, the universe of users consists of the union of all the leaf nodes of the tree with the rest of the nodes being redundant though. While this is a clear downside in terms of storage size the performance of the model should not be affected. The R-Tree structure enforces a minimum occupation on each node preventing the knowledge base to grow dramatically. Thus, while this affects the storage size it won't have a major impact on the performance of the model. Additionally, the proposed models also might suffer from overfitting which is another problem in the area of user modelling. The possibility of overfitting exists as the criterion used for training the model differs from the criterion used for testing. Overfitting occurs due to poor training of the models making the models to memorize from the training sample rather than learning to generalize. While this is a relatively rare problem it might affect the models thus having to train the models from scratch to avoid overfitting.

To conclude and summarize, it can be stated that this thesis makes a great contribution to the area of user modelling establishing a new paradigm of hybrid user models based on experience that can be used across different domains. This new approach also addresses some of the major problems in the area as the scalability problem or the start-up problem. Additionally the thesis contributes to fill an important gap in the literature supplying a standardized framework to evaluate user models irrespectively of their nature and formulation. Thus, user models evaluations are no longer incomparable thanks to this new benchmark.

7 Future works

The first future line involves exploring additional algorithms for both models. Although, RTUM formulation across different functions has provided competitive results along the experimentation the impact of variations on those algorithms is yet to be explored as it is the selection of the best formulation for each domain. This thesis aims to provide an innovative model based on a completely new structure that addresses major problems in the area. However, exploring different algorithms is not the main goal of the thesis while those new formulations can improve the predictions though. In this line, more clustering solutions should be explored, the adapted K-means performed well but other solutions like C-means (Siriporn and Hwajoon, 2009) algorithm might improve the performance for certain domains. In addition, new solutions to replace the fusion procedure or the matching function can also be explored. Ultimately, self-adaptive formulation based on Artificial Intelligence can be seen as the final step in order to get the best formulas for each scenario. Although the self-adaptive formulation is the most appealing solution it is also the most sophisticated one and time consuming.

The second line of work proposed covers studying the impact of new thresholds on RTUM user model. Along the evaluation a set of thresholds has been analysed but whether some other thresholds might improve the performance of the model or not is yet to be analysed. Also regarding the parameterization of the models, along the evaluation some questions came up and should be explored further. While, the presented evaluation provides an extensive study of parameters as K and K_{min} supplementary research can be done in order to better understand the behaviour of the model regarding those parameters.

The third future line comes to alleviate one of the weakness of this proposal as it is the overfitting problem. To that respect techniques to avoid overfitting can be applied to flag when additional training is not resulting in better generalization.

The next future line involves exploring more domains and specifically different typologies of domains. It is certainly true that this thesis explores a good number and typologies of domains but further research can be done in order to determine which domains are easier to predict and which ones are harder. In line with, the domains diversity, features diversity can also be explored. Thus, the impact of biometric features should be studied. Additionally the influence of providing ontology support (offline cleaning) to the features has to be explored. Offline cleaning matches different terms representing the same concept and therefore improves the quality of the domain.

Last but not least, further work can be done in the direction of detecting communities within social domains. This thesis did a first approach in the direction of detecting communities by providing a model that produces a very good clustering to identify those groups within social networks domains. However, this line should be further explored and that clustering should be tested against other methods such as the minimum-cut method or the modularity maximization.

Acknowledgements

This PhD thesis has been supported by the following research projects Thuban (TIN2008-02711), Semants ((TSI-020110- 2009-419)) and Cadooh (TSI-020302-2011-21), supported by the Spanish Ministry of Education and the Spanish Ministry of Industry, Tourism and Commerce.

8 References

- Adomavicius, G. and Tuzhilin, A.: 2001 a, ‘Multidimensional recommender systems: a data warehousing approach’. In Proc. of the 2nd Intl. Workshop on Electronic Commerce (WELCOM’01). Lecture Notes in Computer Science, vol. 2232, Springer.
- Adomavicius, G. and Tuzhilin, A.: 2001 b, ‘Expert-Driven Validation of Rule-Based User Models in Personalization Applications’. Data Min. Knowl. 33-58. DOI=10.1023/A:1009839827683 http://dx.doi.org/10.1023/A:1009839827683.
- Adomavicius, G., Sankaranarayanan, R., Sen, S. and Tuzhilin, A.: 2005, ‘Incorporating contextual information in recommender systems using a multidimensional approach’. ACM Transactions on Information Systems (TOIS), 23(1):103–145.
- Adomavicius, G. and Tuzhilin, A.: 2007, ‘Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions’. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749.
- Adomavicius, G. and Tuzhilin, A.: 2011, ‘Context-Aware Recommender Systems’. In Recommender Systems Handbook. Springer US 217-253. 978-0-387-85820-3
- Aggarwal, C., Joel L., Philip S., Epelman, Y.: 1997, ‘The S-Tree: An Efficient Index for Multidimensional Objects’. In Proceedings of the 5th International Symposium on Advances in Spatial Databases (SSD '97), Michel Scholl and Agnès Voisard (Eds.). Springer-Verlag, London, UK, UK, 350-373.
- Agrawal, R. and Srikant, R.: 1995, ‘Mining sequential patterns’. In Proceedings of the Eleventh International Conference on Data Engineering (ICDE ’95). Taipei, Taiwan, pp. 3–14.
- Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A.I.: 1996, ‘Fast discovery of association rules’. In Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA, Ch. 12.
- Agrawal, R., Rantzau, R. and Terzi, E.: 2006, ‘Context-sensitive ranking’. In Proceedings of the ACM SIGMOD international conference on Management of data, pages 383–394. ACM.
- Ahn, J, J., Byun, H, W., Oh, K, J and Kim, T, Y.: 2012, ‘Bayesian forecaster using class-based optimization’. Applied Intelligence 36, 3, 553-563. DOI=10.1007/s10489-011-0275-2 http://dx.doi.org/10.1007/s10489-011-0275-2
- Akrivas, G., Wallace, M., Andreou, G., Stamou, G. and Kollias, S.: 2002, ‘Context-sensitive semantic query expansion’. In Proceedings of the IEEE international conference on artificial intelligence systems (ICAIS), pages 109–114. Divnomorskoe, Russia.
- Albrecht, D. and Zukerman, I.: 2007, ‘Introduction to the special issue on statistical and probabilistic methods for user modelling’. In User Modeling and User-Adapted Interaction.
- Amento, B., Terveen, L., Hix, D. and Ju, P.: 1999, ‘An empirical evaluation of user interfaces for topic management of web sites’. In Proceedings of the Conference on Human Factors in Computing Systems (CHI ’99). ACM, New York, 552–559.

- Amento, B., Terveen, L., Hill, W., Hix, D. and Schulman, R.: 2003, 'Experiments in social data mining: The TopicShop System'. ACM Trans. Computer-Human Interact. 10, 1 (Mar.), 54–85.
- Anand, S.S., and Mobasher, B.: 2007, 'Contextual recommendation'. WebMine, LNAI, 4737:142–160.
- Andromeda.: 2000, 'LikeMinds'. Andromedia, http://www.andromedia.com/products/likeminds/index.html
- Ankerst, M., Breunig, M.M., Kriegel, H.P. and Sander, J.: 1999, 'OPTICS: ordering points to identify the clustering structure'. In Proceedings of ACM SIGMOD Conference, pp. 49–60.
- Arthur, D. and Vassilvitskii, S.: 2007, 'K-means++: The advantages of careful seeding'. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics Philadelphia, PA, USA, pp. 1027–1035.
- Balabanovic, M. and Shoham, Y.: 1997, 'Fab: Content-based, collaborative recommendation'. Commun. ACM 40, 66–72.
- Baeza-Yates, R., Ribeiro-Neto, B.: 1999, Modern Information Retrieval. Addison-Wesley.
- Basu, C., Hirsh, H. and Cohen W.: 1998, 'Recommendation as Classification: Using Social and Content-Based Information in Recommendation'. In: Proceedings of the 15th National Conference on Artificial Intelligence, Madison, WI, pp. 714-720.
- Bazire, M. and Brézillon, P.: 2005, 'Understanding context before using it'. In A. Dey and et al., editors, Proceedings of the 5th International Conference on Modeling and Using Context. Springer Verlag.
- Beckmann, N. Kriegel, H.P., Schneider, R. and Seeger, B.: 1990, 'The R*-tree: an efficient and robust access method for points and rectangles'. In Proceedings of ACM SIGMOD international conference on Management of data ACM, New York, NY, USA, 322-331.
- Berkovsky, S., Aroyo, L., Heckmann, D., Houben, G.J., Kroner, A., Kuflik, T., Ricci, F.: 2007, 'Providing Context-Aware Personalization through Cross-Context Reasoning of User Modeling Data', UBIDEUM 2007, Proceedings of the International Workshop on Ubiquitous and Decentralized User Modeling, workshop at UM 2007, 11th International Conference on User Modeling, pp. 2-7, Publ. User Modeling Inc.
- Berry, M.J. and Linoff, G.: 1997, 'Data mining techniques: for marketing, sales, and customer support'. John Wiley & Sons, Inc. New York, NY, USA.
- Bessa, A., Laender, A.H.F., Veloso, A. and Ziviani, N.: 2012, ' Alleviating the Sparsity Problem in Recommender Systems by Exploring Underlying User Communities'. CEUR-WS.org, 2012, 866, 35-47.
- Bestavros, A.: 1996, 'Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems'. Proceedings of the International Conference on Data Engineering.
- Bettini, C., Wang, X.S. and Jajodia, S.: 1996, 'Testing complex temporal relationships involving multiple granularities and its application to data mining'. In Proceedings of the

Fifteenth ACM SIGACT-SIGMOD-SIGART. Symposium on Principles of Database Systems (PODS ’96). Montreal, Canada, pp. 68–78.

- Billsus, D. and Pazzani, M.: 1998, ‘Learning collaborative information filters’. In Proceedings of the 15th International Conference on Machine Learning (ICML '98).
- Billsus, D. and M. Pazzani.: 1999, ‘A Hybrid User Model for News Stories Classification’. Procs. of the Seventh International Conference on User Modeling. Banff, Canada, pp 99–108.
- Billsus, D. and Pazzani, M.: 2000, ‘User modelling for adaptive news access’. User Modeling and UserAdapted Interaction, 10(2-3):147-180.
- Billsus, D., Brunk, C.A., Evans, C., Gladish, B. and Pazzani, M.: 2002, ‘Adaptive interfaces for ubiquitous web access’. Commnications of the ACM, 45(5). 34-38.
- Boutemedjet, S. and Ziou, D.: 2008, ‘A graphical model for context-aware visual content recommendation’. IEEE Transactions on Multimedia, 10(1):52–62.
- Brajnik, G. and Tasso, C.: 1994, ‘A shell for developing non-monotonic user modelling systems’. International Journal on Human-Computer Studies 40, pp 31-62.
- Breese, J., Heckerman, D. and Kadie, C.: 1998, ‘Empirical analysis of predictive algorithms for collaborative clustering’. Proc. of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, San Francisco 43.
- Breiman, L., Friedman, J., Olshen, R. and Stone, C.: 1984, ‘Classification and Regression Trees’. Wadsworth International Group.
- Brown, P.J., Bovey, J.D. and Chen, X.: 1997, ‘Context-aware applications: from the laboratory to the marketplace’. IEEE Personal Communications, 4:58–64.
- Buhmann, M.D.: 2001, ‘Approximation and interpolation with radial functions’. In Multivariate Approximation and Applications. Eds. N. Dyn, D. Leviatan, D. Levin, and A. Pinkus. Cambridge University Press.
- Burke, R.: 2002, ‘Hybrid Recommender Systems: Survey and Experiments’. User Modeling and User-Adapted Interaction’. 12, 4. 331-370. http://dx.doi.org/10.1023/A:1021240730564.
- Burke, R.: 2007, ‘Hybrid web recommender systems’. The Adaptive Web, pages 377–408.
- Calle, F.J., Albacete, E., Olaziregi, G., Sánchez, E., Valle Agudo, D., Rivero, J. and Cuadra, D.: 2011, ‘Cognos: A Pragmatic Annotation Toolkit for the Acquisition of Natural Interaction Knowledge’. 27th Conference of the Spanish Society for Natural Language Processing, Huelva (Spain).
- Calle, F.J., Castaño, L., Castro, E., Cuadra, D. (2013). Evaluation framework for statistical user models. 10th International Symposium on Distributed Computing and Artificial Intelligence (DCAI). pp: 449-456, May, Salamanca (Spain).
- Canny, J.: 2002, ‘Collaborative filtering with privacy via factor analysis’. In Proceedings of the 25th Annual International ACMSIGIR Conference on Research and Development in Information retrieval. ACM, New York, 238–245.

- Cantador, I. and Castells, P.: 2009, ‘Semantic contextualization in a news recommender system’. In Workshop on Context-Aware Recommender Systems (CARS 2009). New York.
- Castaño, L., Calle, F.J., Cuadra, D. and Castro, E.: 2011, ‘User Modeling for Human-Like Interaction’. The 2nd International Workshop on User Modeling and Adaptation for Daily Routines (UMADR), pp 23-34.
- Cena, F., Console, L., Gena, C., Goy, A., Levi, G., Modeo, S. and Torre, I.: 2006, ‘Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide’. AI Communications, 19(4). 369–384.
- Chin D.: 2001, ‘Empirical Evaluation of User Models and User-Adapted Systems’. User Modeling and User-Adapted Interaction 181-194.
- Chiu, B.C. and Webb, G.: 1998, ‘Using Decision Trees for Agent Modeling: Improving Prediction Performance’. User Modeling and User-Adapted Interaction 8(1-2), pp 131-152.
- Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: 1999, ‘Combining Content-Based and Collaborative Filters in an Online Newspaper’. SIGIR ’99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA. http://www.cs.umbc.edu/~ian/sigir99rec/papers/claypool_m.ps.gz
- Cleverdon, C. and Kean, M.: 1968, ‘Factors Determining the Performance of Indexing Systems’. Aslib Cranfield Research Project, Cranfield, England.
- Cohen, W.W., Schapire, R.E. and Singer, Y.: 1999, ‘Learning to order things. Journal of Articial Intelligence Research’, 10:243-270.
- Comer, D.: 1979, ‘Ubiquitous B-Tree’. ACM Comput. Surv. 11, 2, 121-137. DOI=10.1145/356770.356776 http://doi.acm.org/10.1145/356770.356776
- Cortes, C., Fisher, K., Pregibon, D. and Rogers, A.: 2000, ‘Hancock: a language for extracting signatures from data streams’. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining . ACM, New York, NY, USA, 9-17.
- Cosley, D., Lam, S.K., Albert, I., Konstan, J.A. and Riedl, J.: 2003, ‘Is seeing believing? How recommender interfaces affect users’ opinions’. CHI Lett. 5.
- Dahlen, B.J., Konstan, J A., Herlocker, J.L., Good, N., Borchers, A., and Riedl, J.: 1998, ‘Jumpstarting Movielens: User benefits of starting a collaborative filtering system with dead data’. TR 98-017. University of Minnesota.
- Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R.: 1990, ‘Indexing by latent semantic analysis’. Journal of the American Society for Information Science, vol. 41, no. 6, pp. 391–407.
- Deshpande, M. and Karypis, G.: 2004, ‘Item-based top-N recommendation algorithms’. ACM Transactions on Information Systems, vol. 22, no. 1, pp. 143–177.
- Dey, A.K., Abowd, G.D. and Salber, D.: 2001, ‘A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications’. Human-Computer Interaction, 16(2):97–166.

- Domingos, P., and Richardson, M.: 2003, ‘Mining the network value of customers’. In Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining. ACM, New York, 57–66.
- Dourish, P.: 2004, ‘What we talk about when we talk about context’. Personal and ubiquitous computing, 8(1):19–30.
- Duchon, J.: 1979, ‘Splines minimizing rotation-invariate semi-norms in Sobolev spaces’. In Constructive Theory of Functions of Several Variables, ed. W. Schempp and Zeller, pp. 85-100, Springer.
- Elmasri, R. and Navathe. S.B.: 2006, ‘Fundamentals of Database Systems (5th Edition)’. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Ester, M., Kriegel, H.P., Sander, J. and Xu, X.: 1996, ‘A density-based algorithm for discovering clusters in large spatial databases with noise’. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD '96).
- Finin, T. and Drager, D.: GUMS: 1986, ‘A general user modelling system’. In Proceedings of the workshop on Strategic computing natural language. Association for Computational Linguistics, Stroudsburg, PA, USA, pp 224-230.
- Finin, T.W.: 1989, ‘GUMS: A general user modelling shell’. In: A. Kobsa and W. Wahlster (eds.), User Models in Dialog Systems. Springer-Verlag, Berlin, Heidelberg, pp. 411-430.
- Fix, E., Hodges, J.L.: 1951, ‘Discriminatory analysis, nonparametric discrimination: Consistency properties’. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas.
- Freeston, M.W.: 1987 ‘The BANG file: a new kind of grid file’ Proc. ACM Sigmod International Conference on the Management of Data, San Fran-cisco, 260-269.
- Fu, Y.C., Hu, Z.Y., Guo, W., Zhou, D.R.: 2003, ‘QR-tree: a hybrid spatial index structure’. Machine Learning and Cybernetics, International Conference on , vol.1, no., pp.459,463 Vol.1, 2-5.
- Ghazanfar, M.A. and Prugel-Bennett, A.: 2011, ‘Fulfilling the Needs of Gray-Sheep Users in Recommender Systems, A Clustering Solution’. In, International Conference on Information Systems and Computational Intelligence, Harbin, China, 18 – 20.
- Goldberg, D., Nichols, D., Oki, B.M. AND Terry, D.: 1992, ‘Using collaborative filtering to weave an information tapestry’. Commun. ACM 35, 61–70.
- Goldberg, K., Roeder, T., Guptra, D. and Perkins, C.: 2001, ‘Eigentaste: A constant-time collaborative filtering algorithm’. Inf. Retr. 4, 133–151.
- Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B.M., Herlocker, J.L. and Riedl, J.: 1999, ‘Combining collaborative filtering with personal agents for better recommendations’. In Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99), Eds. AAAI Press, Menlo Park, Calif., 439–446.
- Groenen, P.J.F., Kaymak, U. and van Rosmalen, J.: 2007, ‘Fuzzy Clustering with Minkowski Distance Functions’. In Advances in Fuzzy Clustering and its Applications 10.1002/9780470061190.ch3. http://repub.eur.nl/res/pub/7873/ei2006-24.pdf.

- Günther, O.: 1989. 'The Design of the Cell Tree: An Object-Oriented Index Structure for Geometric Databases'. In Proceedings of the Fifth International Conference on Data Engineering. IEEE Computer Society, Washington, DC, USA, 598-605.
- Güting, R.H., de Almeida, V.T., Ansorge, D., Behr, T., Ding, Z., Höse, T. Hoffmann, F., Spiekermann, M. and Telle, U.: 2005, 'SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching'. In Proceedings of the International Conference on Data Engineering, ICDE.
- Guttman, A.: 1984, 'R-trees: A dynamic index structure for spatial searching'. ACM SIGMOD Int. Conference on Management of Data. ACM, New York, NY, USA, pp 47-57.
- Han, J. and Kamber, M.: 2001, 'Data Mining: Concepts and Techniques'. San Francisco, California, USA.
- Harter, S.P.: 1996, 'Variations in relevance assessments and the measurement of retrieval effectiveness'. J. ASIS 47, 37–49.
- Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R. and Kadie, C.: 2000, 'Dependency networks for inference, collaborative filtering, and data visualization'. J. Mach. Learn. Res. 1, 49–75.
- Heckmann, D., Schwartz, T., Brandherm, B., & Schmitz, M., & von Wilamowitz-Moellendorff, M.: 2005. GUMO - the General User Model Ontology. Proceedings of the 10th International Conference on User Modeling UM '05, LNCS, (Vol. 3538, pp. 428–432). Springer.
- Herlocker, J., Konstan, J., Borchers, A. and Riedl, J.: 1999, 'An algorithmic framework for performing collaborative clustering'. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, pp 230-237.
- Herlocker, J.L., Konstan, J.A. and Riedl, J.: 2000, 'Explaining collaborative filtering recommendations'. In Proceedings of the 2000 Conference on Computer Supported Cooperative Work, 241–250.
- Herlocker, J.L. and Konstan, J.A.: 2001, 'Content-independent task-focused recommendation'. IEEE. Internet Computing, pages 40–47.
- Herlocker, J.L., Konstan, J.A. and Riedl, J.: 2002, 'An empirical analysis of design choices in neighbourhood-based collaborative filtering algorithms'. Inf. Retr. 5, 287–310.
- Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T.: 2004 'Evaluating collaborative filtering recommender systems'. ACM Transactions on Information Systems, vol. 22, no. 1, pp. 5–53.
- Hill, W., Stead, L., Rosenstein, M. and Furnas, G.W.: 1995, 'Recommending and evaluating choices in a virtual community of use'. In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. ACM, New York, 194–201.
- Hoagland, A.S.: 1972 'Mass storage: past, present and future'. In Proceedings of the December 5-7, fall joint computer conference, part II (AFIPS '72 (Fall, part II)). ACM, New York, NY, USA, 985-991.

- Horvitz, E.: 1998, ‘Continual Computation Policies for Utility-Directed Prefetching’. CIKM Proceedings of the Seventh ACM Conference on Information and Knowledge Management. Bethesda, Maryland, pp 175-184.
- Jagadish, H.V.: 1990. ‘Linear clustering of ob jects with multiple attributes’. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 332-342.
- Jennings, A. and Higuchi, H.: 1993, ‘A User Model Neural Network for a Personal News Service’. User Modeling and User-Adapted Interaction 3, pp 1-25.
- Jin, R., Si, L. and Zhai, C.: 2003, ‘Preference-based Graphic Models for Collaborative Filtering’. In Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003), Acapulco, Mexico.
- Joerding, T.: 1999, ‘Temporary UserModeling for Adaptive Product Presentation in theWeb’. Procs. of the Seventh International Conference on User Modeling. Banff, Canada, pp 333-334.
- Jones, S.K.: 1972, ‘A statistical interpretation of term specificity and its applications in retrieval’. Journal of Documentation, vol. 28, no. 1, pp. 11–21.
- Jones, G.J.F., Glasnevin, D. and Gareth, I.: 2005, ‘Challenges and opportunities of context aware information access’. In International Workshop on Ubiquitous Data Management, pages 53–62.
- Karypis, G.: 2001, ‘Evaluation of item-based top-N recommendation algorithms’. In Proceedings of the International Conference on Information and Knowledge Management (CIKM '01), pp. 247–254, Atlanta, Ga, USA.
- Kautz, H.: 1998, ‘Recommender systems’. Technical Report WS-98-08. AAAI Press.
- Kendall, M.: 1938, ‘A New Measure of Rank Correlation’. Biometrika 30 (1–2): 81–89. DOI:10.1093/biomet/30.1-2.81. JSTOR 2332226.
- Kitts, B., Freed, D. and Vrieze, M.: 2000, ‘Cross-sell: A fast promotion-tunable customer–item recommendation method based on conditional independent probabilities’. In Proceedings of ACM SIGKDD International Conference. , ACM, New York, 437–446.
- Kobsa, A.: 2001, ‘Generic User Modeling Systems’. User Modeling and User-Adapted Interaction 11, 1-2, pp 49-63.
- Kobsa, A. and Pohl, W.: 2001, ‘The user modelling shell system BGP-MS’. User Modeling and User-Adapted Interaction 4, pp 59-106.
- Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedl, J.: 1997. ‘GroupLens: Applying Collaborative Filtering to Usenet News’. Communications of the ACM, March, 40(3): 77-87.
- Konstan, J.A., Riedl, J., Borchers, A. and Herlocker, J.L.: 1998, ‘Recommender systems: a GroupLens perspective’. In Recommender Systems. Papers Workshop. Technical Report WS-98-08. AAAI Press.
- Koren, Y.: 2008, ‘Factorization meets the neighbourhood: a multifaceted collaborative filtering model’. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 426–434. ACM, New York.
- Krulwich, B.: 1997, ‘Lifestyle finder: intelligent user profiling using large-scale demographic data’. Artificial Intelligence Magazine, vol. 18, no. 2, pp. 37–45.

- Lang, K.: 1995, 'Nesweeder: Learning to filter netnews'. In Proceedings of the 12th International Conference on Machine Learning.
- Lau, T. and Horvitz, E.: 1999, 'Patterns of Search: Analyzing and Modeling Web Query Refinement. In: UM99 Proceedings of the Seventh International Conference on User Modeling. Banff, Canada, pp. 119–128.
- Landauer, T. and Littman, M.: 1994, 'Computerized cross-language docu- ment retrieval using latent semantic indexing'. US patent no. 5301109.
- Linden, G., Smith, B. and York, J.: 2003, 'Amazon.com Recommendations: Item-to-Item Collaborative Filtering'. IEEE Internet Computing, Jan.-Feb.
- Littlestone, N. and Warmuth, M.: 1994, 'The Weighted Majority Algorithm. Information and Computation'. 108(2):212-261.
- Maamar, Z., Benslimane, D. and Narendra, N.C.: 2006, 'What can context do for web services?'. Communications of the ACM, 49(12):98–103.
- MacQueen, J.B.: 1967, 'Some methods for classification and analysis of multivariate observations'. In Proceedings of the 5th Symposium on Math, Statistics, and Probability, pp. 281–297, Berkeley, California, USA.
- Mannila, H. and Toivonen, H.: 1996, 'Discovering generalized episodes using minimal occurrences'. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. Portland, OR, pp. 146–151.
- Mannila, H., Toivonen, H. and Verkamo, A.I.: 1997, 'Discovery of Frequent Episodes in Event Sequences'. Data Min. Knowl. Discov. 1, 3, 259-289.
- McCrae, J., Piatek, A. and Langley, A.: 2004, 'Collaborative Filtering'. http://www.imperialviolet.org/.
- McDonald, D.W.: 2001, 'Evaluating Expertise Recommendations'. In Proceedings of the ACM2001 International Conference on Supporting Group Work (GROUP'01). ACM, New York.
- Miller, B., Albert, I., Lam, S.K., Konstan, J. and Riedl, J.: 2003, 'MovieLens Unplugged: Experiences with a Recommender System on Four Mobile Devices'. Proceedings of the 17th Annual Human-Computer Interaction Conference (HCI 2003), British HCI Group, September.
- Miyahara, K. and Pazzani, M.J.: 2002, 'Improvement of collaborative filtering with the simple Bayesian classifier'. Information Processing Society of Japan, vol. 43, no. 11.
- Mokbel, M.F., Xiong, X., Aref, W.G., Hambrusch, S., Prabhakar, S. and Hammad, M.: 2004, 'PLACE: A Query Processor for Handling Real-time Spatio-temporal Data Streams (Demo)'. In Proceedings of the International Conference on Very Large Data Bases, VLDB.
- Mokbel, M.F. and Levandoski, J.J.: 2009, 'Toward context and preference-aware location-based services'. In Proceedings of the Eighth ACM International Workshop on Data Engineering for Wireless and Mobile Access, pages 25–32. ACM.
- Mooney, R. J. and Roy, L.: 1999, 'Content-Based Book Recommending Using Learning for Text Categorization'. SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA. http://www.cs.umbc.edu/~ian/sigir99-rec/papers/mooney_r.ps.gz

- Morales, R. and Pain, H.: 1999, ‘Modeling of Novices' Control Skills with Machine Learning’. Procs. of the Seventh International Conference on User Modeling. Banff, Canada, pp 159-168.
- Mustansar, A. and Prügel-Bennett, A.: 2014. ‘Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems’. Expert Syst. Appl. 41, 7 (June 2014), 3261-3275.
- Newman, W.: 1997, ‘Better or just different? On the benefits of designing interactive systems in terms of critical parameters’. In Proceedings of the Designing Interactive Systems. ACM, New York, 239–246.
- Nichols, D.M.: 1997, ‘Implicit Ratings and Riltering’. In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering, 10-12. Budapaest, Hungary, ERCIM.
- Nurnberger, G.: 1989, ‘ Approximation by Spline Functions’. Springer-Verlag.
- Oard, D.W. and Kim, J.: 1998, ‘Implicit feedback for recommender systems’. In Recommender Systems. Technical Report WS-98-08. AAAI Press.
- O'Connor, M. and Herlocker, J.: 1999, ‘Clustering items for collaborative filtering’. In Proceedings of the ACM SIGIR Workshop on Recommender Systems.
- Oku, K., Nakajima, S., Miyazaki, J. and Uemura, S.: 2006, ‘Context-aware SVM for context-dependent information recommendation’. In Proceedings of the 7th International Conference on Mobile Data Management, page 109.
- Oosterom, P.: 1999 ‘Spatial Access Methods’. Geographical Information Systems Vol.1 385-400. John Wiley.
- Orwant, J.: 1995, ‘Heterogenous learning in the Doppelganger user modelling system’. User Modeling and User-Adapted Interaction 4(2), pp 107-130.
- Padmanabhan, B. and Tuzhilin, A.: 1999, ‘Unexpectedness as a measure of interestingness in knowledge discovery’. Decision Support Systems, 27(3):303–318.
- Paiva, A. and Self, J.: 1994, ‘TAGUS a user and learner modelling workbench’. User Modeling and User-Adapted Interaction 4(3), pp 197–228.
- Palmisano, C., Tuzhilin, A. and Gorgoglione, M.: 2008, ‘Using context to improve predictive modelling of customers in personalization applications’. IEEE Transactions on Knowledge and Data Engineering, 20(11):1535–1549.
- Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C. and Pedone, A.: 2009, ‘Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems’. In Proceedings of the 3rd ACM conference on Recommender systems, pages 265–268. ACM.
- Papadias, D. and Theodoridis, Y.: 1997, ‘Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures’. In Proceedings of International Journal of Geographical Information Science.111-138.
- Pazzani, M. and Billsus, D.: 1997, ‘Learning and revising user profiles: the identification of interesting web sites’. Machine Learning, vol. 27, no. 3, pp. 313–331.
- Pazzani, M.: 1999, ‘A framework for collaborative, content-based and demographic filtering’. Artificial Intelligence Review, pages 393-408.

- Pearson, K.: 1896, ‘Mathematical contributions to the theory of evolution. III. Regression, heredity and panmixia’. Philos. Trans. Royal Soc. London Ser. A , 187. pp. 253–318.
- Peddy, C.C. and Armentrout, D.: 2003, ‘Building Solutions with Microsoft Commerce Server 2002’. Microsoft Press.
- Pennock, D.M. and Horvitz, E.: 1999, ‘Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach’. In IJCAI'99 Workshop: Machine Learning for Information Filtering.
- Perrault, C.R., Allen, J.F. and Cohen, P.R.: 1978, ‘Speech acts as a basis for understanding dialogue coherence’. Report 78-5, Department of Computer Science, University of Toronto, Canada.
- Peppers, D. and Rogers, M.: 1993, ‘The One-to-One Future’. Doubleday, New York, NY.
- Piatetsky-Shapiro, G. and Matheus, C.J.: 1994, ‘The interestingness of deviations’. In Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases.
- Popescul, A., Ungar, L.H., Pennock, D.M. and Lawrence, S.: 2001, ‘Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments’. In Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI '01), pp. 437–444.
- Prahalad, C.K.: 2004, ‘Beyond CRM: CK Prahalad predicts customer context is the next big thing’. American Management Association MwWorld.
- Quinlan, J.: 1993, ‘C4.5: Programs for Machine Learning’. Morgan Kaufmann.
- Ramakrishnan, R. and Gehrke, J.: 2000, ‘Database Management Systems (2nd ed.)’. Osborne/McGraw-Hill, Berkeley, CA, USA.
- Resnick, P., Iakovou, N., Suchak, M., Bergstrom, P., and Riedl, J.: 1994, ‘GroupLens: An open architecture for collaborative filtering of net news’. In Proceedings of the 1994 Conference on Computer. Supported Collaborative Work. R. Furuta and C. Neuwirth, Eds. ACM, New York. 175–186.
- Ricci, F. and Nguyen, Q.N.: 2006, ‘Mobyrek: A conversational recommender system for on-the-move travelers’. Destination Recommendation Systems: Behavioural Foundations and Applications, pages 281–294.
- Rich, E.: 1979, ‘User modelling via stereotypes’. Cognitive Science, Vol. 3, No. 4, pp 329-354.
- Rocchio, Jr.J.: 1971, ‘Relevance feedback in information retrieval’. In The SMART System – Experiments in Automatic Document Processing, New York: Prentice Hall, pp. 313-323. 4, ‘Automatic Text Processing’. Addison-Wesley.
- Rodden, T., Cheverst, K., Davies, K. and Dix, A.: 1998, ‘Exploiting context in hci design for mobile systems’. In Workshop on Human Computer Interaction with Mobile Devices, pages 21–22.
- Ryan, N., Pascoe, J. and Morse, D.: 1997, ‘Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant’. Computer Applications in Archaeology. British Archaeological Reports, Oxford.
- Salton, G. and McGill, M.: 1983, ‘Introduction to Modern Information Retrieval’. McGraw-Hill, New York, NY, USA.

- Sarwar, B.M., Karypis, G.J., Konstan, A. and Riedl, J.: 2000, 'Analysis of recommendation algorithms for E-commerce'. In Proceedings of the ACM E-Commerce, pp. 158–167, Minneapolis.
- Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J.: 2001, 'Item-based collaborative filtering recommendation algorithms'. In Proceedings of the 10th International Conference on World Wide Web (WWW '01), pp. 285–295.
- Schaback, R. and Wendland, H.: 2001, 'Characterization and construction of radial basis functions'. In Multivariate Approximation and Applications. Eds. N. Dyn, D. Leviatan, D. Levin and A. Pinkus. Cambridge University Press.
- Shardanand, U. and Maes, P.: 1995, 'Social information filtering: Algorithms for automating word of mouth'. In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. ACM, New York. 210–217.
- Schein, A.I., Popescul, A., Ungar, L.H., and Pennock, D.M.: 2001, 'Generate models for coldstart recommendations'. Proceedings of the ACMSIGIR Workshop on Recommender Systems. ACM, New York.
- Schilit, B.N. and Theimer, M.M.: 1994, 'Disseminating active map information to mobile hosts'. IEEE network, 8(5): 22–32.
- Schiller, J.H. and Voisard A.: 2004, 'Location-based services'. Morgan Kaufmann.
- Sheth, B. and Maes, P.: 1993, 'Evolving agents for personalized information filtering'. In Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications.
- Sieg, A., Mobasher, B. and Burke, R.: 2007, 'Representing context in web search with ontological user profiles'. In Proceedings of the 6th International Conference on Modeling and Using Context.
- Siriporn, S., Hwajoon, K.: 2009, 'Modified fuzzy ants clustering approach'. Applied Intelligence 31, 2, 122-134. http://dx.doi.org/10.1007/s10489-008-0117-z
- Smyth, B. and Cotter, P.: 2000, 'A Personalized TV Listings Service for the Digital TV Age'. Knowledge-Based Systems 13: 53-59.
- Soboroff, I., Nicholas, C. and Pazzani, M.J.: 1999, ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation.
- Spearman, C.: 1904, 'The proof and measurement of association between two things'. Amer. J. Psychol. 15: 72–101.
- Stanley, K.O.: 2003, 'Learning concept drift with a committee of decision trees'. Tech. Report UTAITR-03-302, Department of Computer Sciences, University of Texas at Austin, USA.
- Stefanidis, K., Pitoura, E. and Vassiliadis, P.: 2007, 'A context-aware preference database system'. International Journal of Pervasive Computing and Communications, 3(4):439–600.
- Stevens, C.: 1993, 'Knowledge-Based Assistance for Ac- cessing Large, Poorly Structured Information Spaces'. Ph.D. dissertation, Dept. of Computer Science, Univ. of Colorado, Boulder.
- Swearingen, K. and Sinha, R.: 2001, 'Beyond algorithms: An HCI perspective on recommender systems. In Proceedings of the SIGIR 2001 Workshop on Recommender Systems.

- Swets, J.A.: 1969, ‘Effectiveness of information retrieval methods’. Amer. Doc. 20, 72–89.
- Tamminen, M.: 1982, ‘The extendible cell method for closest point prob-lems’ BIT. Vol. 22, 27-41
- Timos K, Roussopoulos, N and Faloutsos, C.: 1987, ‘The R+-Tree: A Dynamic Index for Multi-Dimensional Objects’. In Proceedings of the 13th International Conference on Very Large Data Bases (VLDB '87), Peter M. Stocker, William Kent, and Peter Hammersley (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 507-518.
- Tran, T. and Cohen, R.: 2000, ‘Hybrid Recommender Systems for Electronic Commerce’. In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, AAAI Technical Report WS-00-04. pp. 78-83. Menlo Park, CA: AAAI Press.
- Tsymbal, A.: 2004, ‘The problem of concept drift: Definitions and related work’. Technical Report. Department of Computer Science, Trinity College: Dublin, Ireland.
- Turpin, A. and Hersh, W.: 2001, ‘Why batch and user evaluations do not give the same results’. In Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, 17–24.
- Van Setten, M., Pokraev, S. and Koolwaaij, J.: 2004, ‘Context-aware recommendations in the mobile tourist application compass’. In W. Nejdl and P. De Bra, editors, Adaptive Hypermedia, pages 235–244. Springer Verlag.
- Vergara, H.: 1994, ‘PROTUM A Prolog based tool for user modelling’. WIS Memo 10. Univ. of Kostanz, Germany.
- Wade, W.: 2003, ‘A grocery cart that holds bread, butter and preferences’. NY Times, Jan. 16, 2003.
- Wang, W., Yang, J. and Richard R.M.: 1997, ‘STING: A Statistical Information Grid Approach to Spatial Data Mining’. In Proceedings of the 23rd International Conference on Very Large Data Bases. San Francisco, CA, USA, 186-195.
- Widmer, G. and Kubat, M.: 1993, ‘Effective learning in dynamic environments by explicit context tracking’. Proc. 6th European Conference on Machine Learning ECML, Springer-Verlag, Lecture Notes in Computer Science 667, 1993, 227-243.
- Widmer, G. and Kubat, M.: 1996, ‘Learning in the presence of concept drift and hidden contexts’. European Conference on Machine Learning. 23 (1), 69-101.
- Willmott, C.J. and Matsuura, K.: 2005, ‘Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance’. Climate Research 30: 79–82.
- Wittig, F. and Jameson, A.: 2000, ‘Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks’. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp 644–652.
- Woerndl, W., Schueller, C. and Wojtech, R.: 2007, ‘A hybrid recommender system for context-aware recommendations of mobile applications’. In Proceedings of the 3rd International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces, pages 871–878.

- Xiaoyuan, Su. and Taghi, M.: 2009, ‘A Survey of Collaborative Filtering Techniques’. Advances in Artificial Intelligence. Article ID 421425, 19.
- Yao, Y.Y.: 1995, ‘Measuring retrieval effectiveness based on user preference of documents’. J. ASIS. 46, 133–145.
- Yu, K., Schwaighofer, A., Tresp, V., Xu, X. and Kriegel, H.P.: 2004, ‘Probabilistic memory-based collaborative filtering’, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 1, pp. 56–69.
- Yu, Z., Zhou, X., Zhang, D., Chin, C.Y., Wang, X. and Men, J.: 2006, ‘Supporting context-aware media recommendations for smart phones’. IEEE Pervasive Computing, 5(3):68–75.
- Zanker, M., Jessenitschnig, M., Jannach, D. and Gordea, S.: 2007, ‘Comparing recommendation strategies in a commercial context’. IEEE Intelligent Systems, Special issue on Recommender Systems, Vol. 22(3), pp. 69-73.
- Zhang, T., Ramakrishnan, R. and Livny, M.: 1996, ‘BIRCH: an efficient data clustering method for very large databases’. In Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 25, pp. 103–114, Montreal, Canada.
- Zhang, Y., Callan, J. and Minka, T.: 2002, ‘Novelty and redundancy detection in adaptive filtering’. In Proceedings of the 25th Annual International ACM SIGIR Conference, pp. 81-88.
- Zhao, J. and Bose, B.K., 2002. ‘Evaluation of membership functions for fuzzy logic controlled induction motor drive’. In IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the](Vol. 1, pp. 229-234).
- Ziegler, C.N., McNee, S.M., Konstan, J.A. and Lausen, G.: 2005, ‘Improving recommendation lists through topic diversification’. In Proceedings of the 14th international conference on World Wide Web, pages 22–32. Chiba, Japan.
- Zukerman, I., Albrecht, D.W., and Nicholson, A.E.: 1999, ‘Predicting Users' Requests on the WWW’. UM99 Proceedings of the Seventh International Conference on User Modeling. Banff, Canada, pp 275-284.
- Zukerman, I. and Albretch, W.: 2001, ‘Predictive Statistical Models for User Modeling’. User Modeling and User-Adapted Interaction 11, pp 1-2.

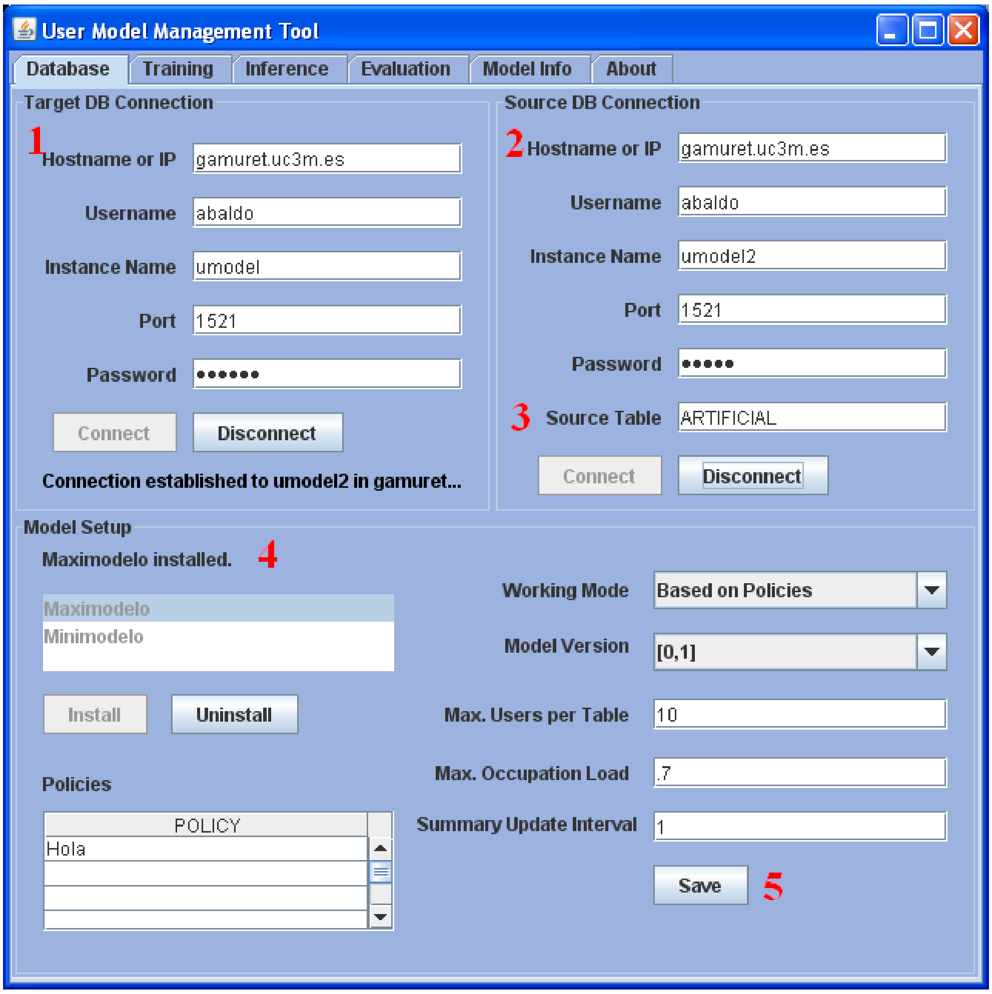
UMTool user manual

Application tabs

1. Connection to databases
2. Training
3. Inference
4. Evaluation
5. Monitoring
6. About
7. Console

Connection to databases

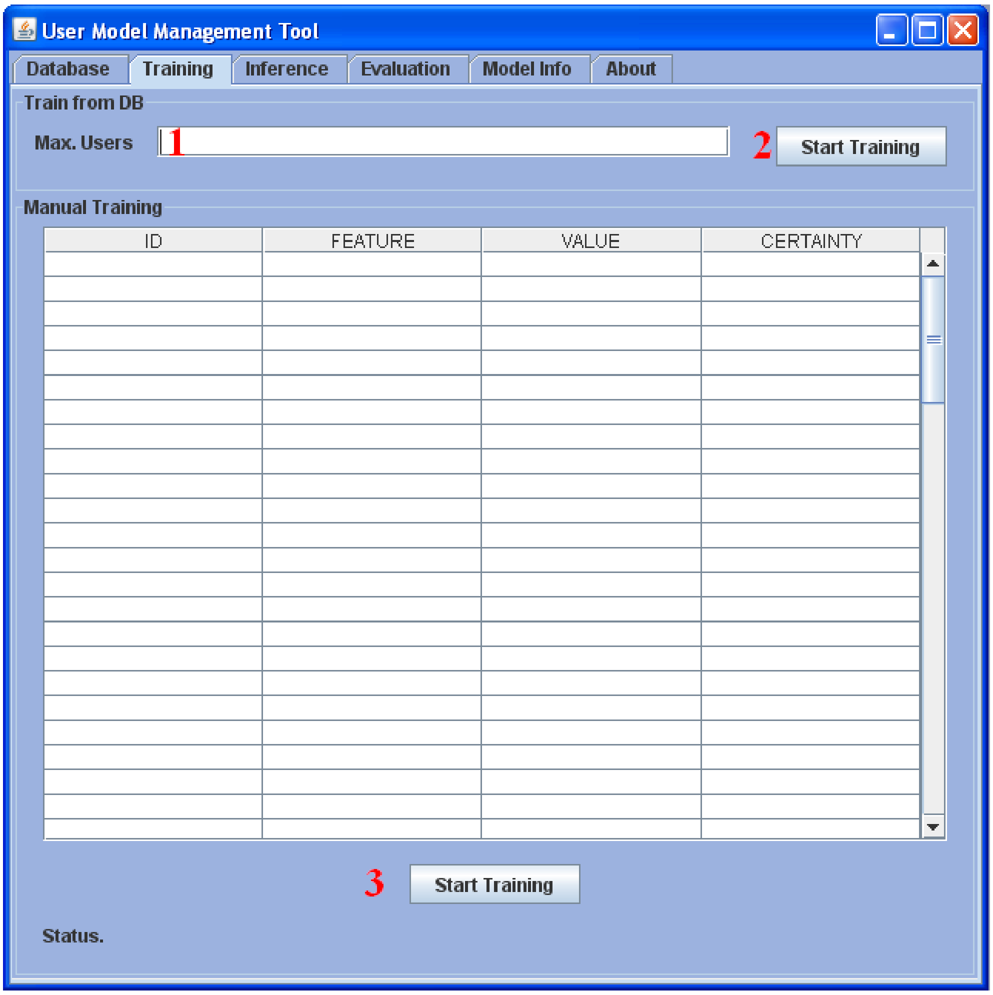
Figure 102 Connection tab



1. Target DB connection: enables a connection to the model instance.
2. Source DB connection: enables a second connection to the raw data.
3. Source table: table with source data.
4. Model Setup: Checks whether some model is installed, enables configuration.
5. Save: Stores permanently the model configuration.

Model training

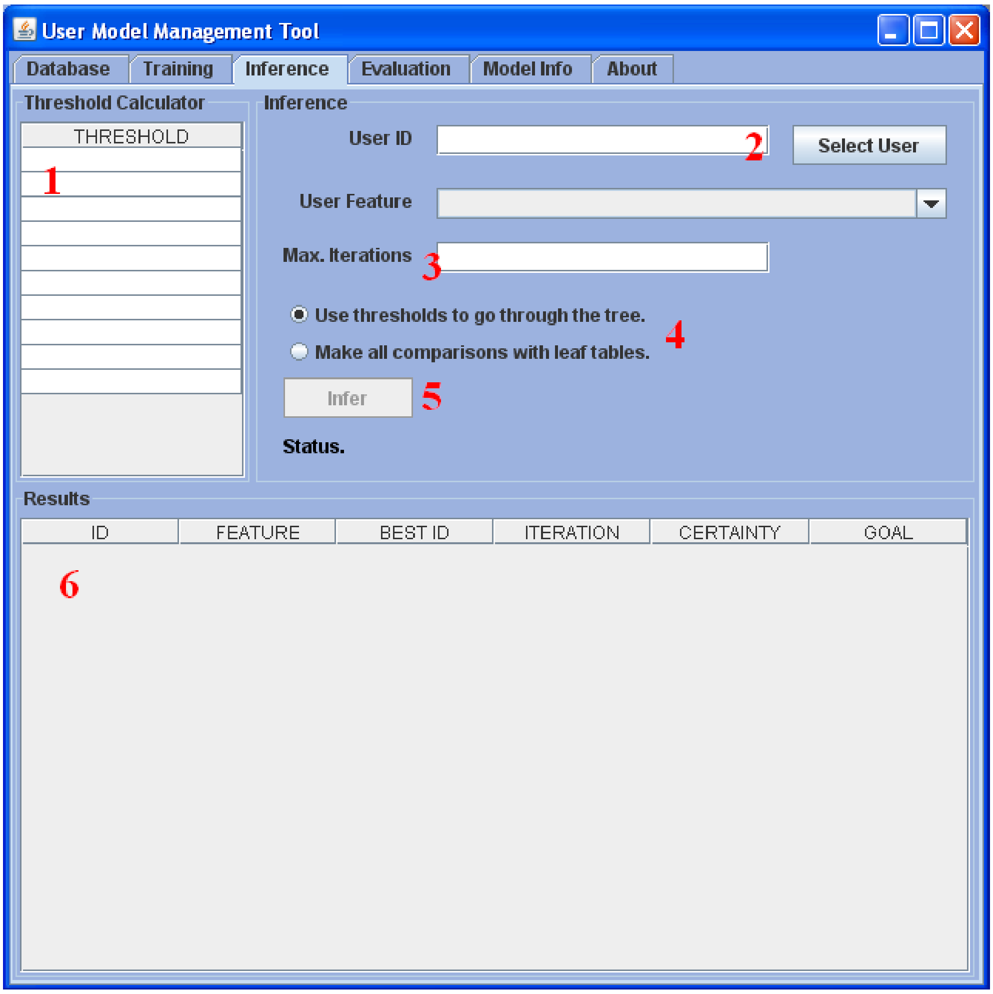
Figure 103 Training tab



1. Max users: Number of user to achieve the model training
2. Automatic training: Enables training since the source database
3. Manual training: Enables inserting users manually.

Model Inference

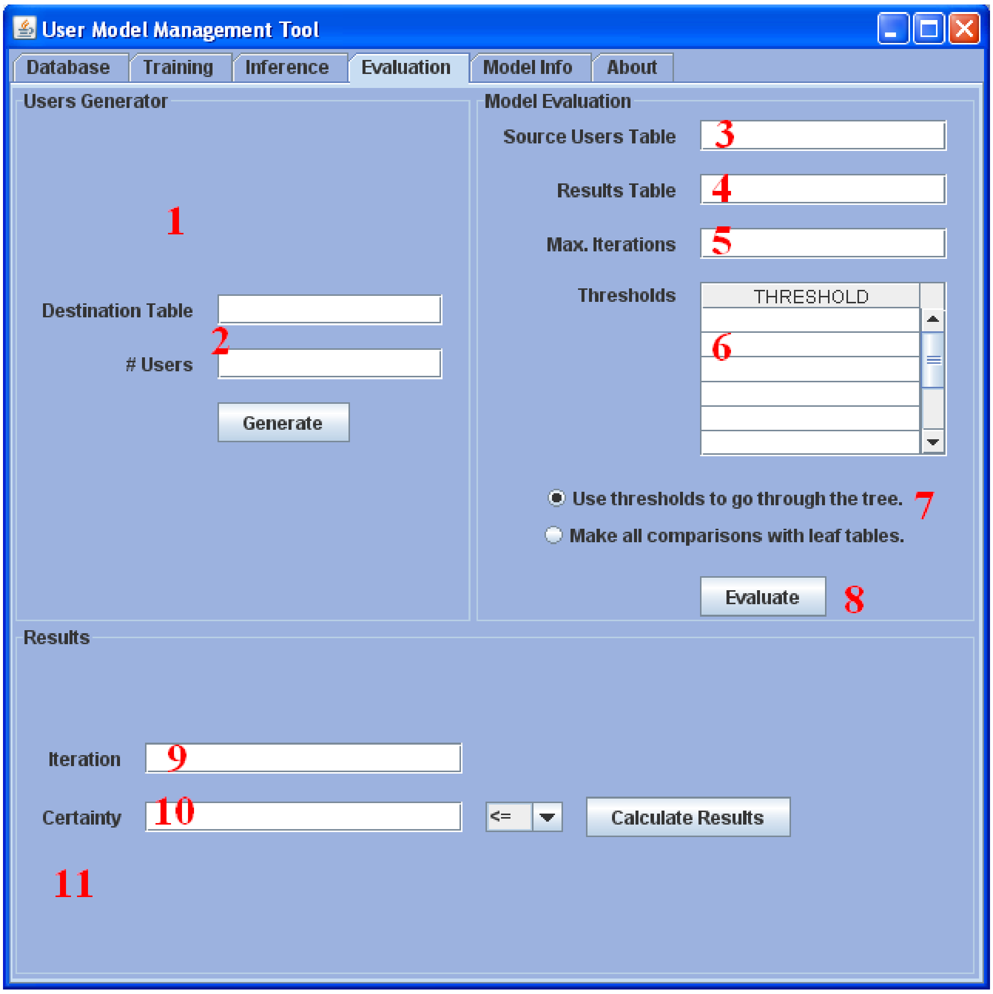
Figure 104 Inference tab



1. Threshold table: Thresholds for the inference. (Use only with "Use thresholds to go through the tree").
2. User: Selects a user and depicts the user in "User feature".
3. Max Iterations: Maximum iterations for the inference.
4. Mode: Enables changing the inference method, using thresholds or inference on the leaves.
5. Infer: Starts the inference.
6. Result: Shows inference results.

Model Evaluation

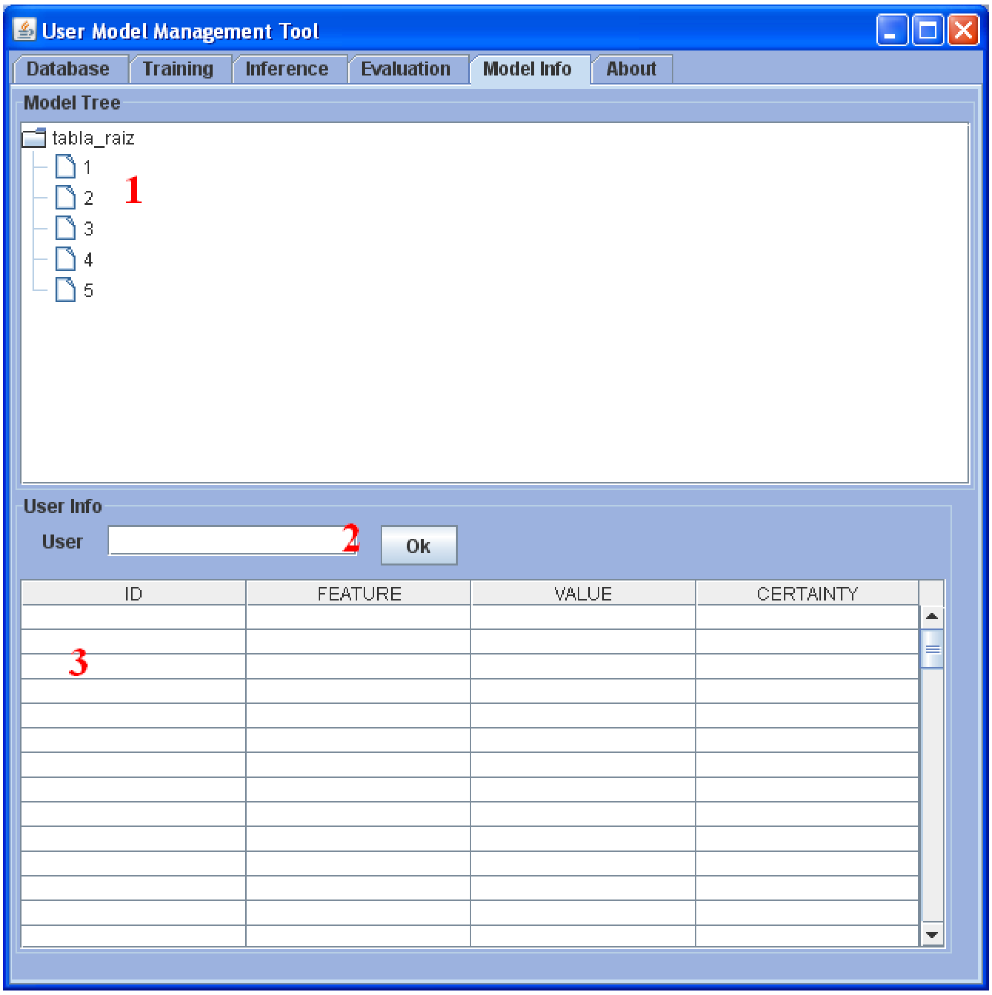
Figure 105 Evaluation tab



1. User's generator: Selects the source data connection, "n" random users and fix a random feature.
2. Destination table: Determines the table where users are stored. If the table does not exists, creates a new one.
3. Source users table: Table set in 2.
4. Result table: Table to store the results yielded by the evaluation. If the table does not exists, creates a new one.
5. Max iterations: Maximum iterations for the evaluation.
6. Thresholds: Value for the different thresholds starting in the tree level 1 and until the level n (leaves). If thresholds are missing evaluation is done on the root.
7. Mode: Enables using thresholds or doing evaluation on the leaves.

Monitoring

Figure 106 Monitoring tab



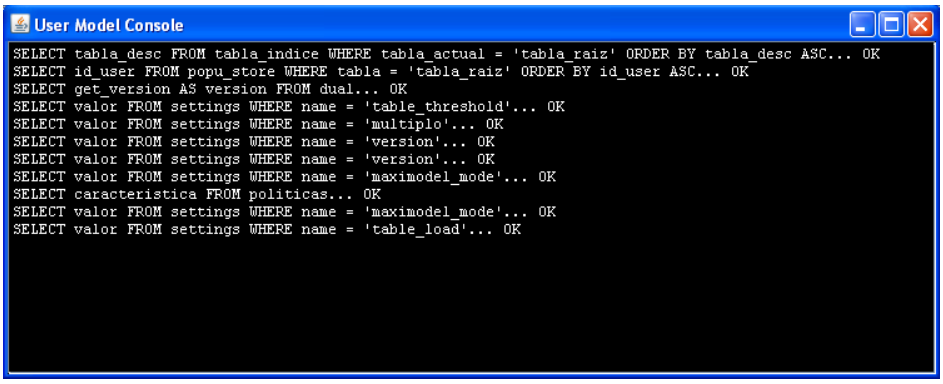
1. Model tree: Displays the tree structure. Enables visiting some user or some table of the tree structure.
2. User info: Enables searching by user identifier. Also shows the path in the tree structure to reach that user.
3. Info: Shows model data.

About

Information about tool developers

Application console

Figure 107 Console



The application employs one or more consoles to show all the instructions that are executed in real time. Errors that happen during the execution will be displayed in red.

Glossary

- Predictive: That is has the ability to infer implicit information.
- Statistical: Based on statistical formulation
- Content-based: Based on your own behaviour
- Collaborative: Based on other like-minded people
- Probability: Frequency of a result when performing some experiment
- Certainty: Quality of truth
- Disk: Permanent storage medium. It contains sectors of some block size, which is the minimum accessible unit.
- Block: The smallest writable and readable unit. Every operation in a file system is done over blocks.
- Partition: Subset of blocks in a disk. A disk may have several partitions.
- Volume: Collection of blocks on some storage medium. A volume might me some portion of the blocks of some disk but it can also cover blocks of several disks.
- Superblock: Part of a volume where the most critical information is stored. For example, the name and the length of the volume.
- Metadata: General term used to refer to information about something but not directly part of it. For example the name of a file is important information but is not part of the file.
- I-node: Place where metadata of a file is stored. I-nodes also contain pointers to the contents of the file.
- Extent: Also known as block runs, extents are a starting block number and the length of the extent. Extents are always contiguous.
- Attribute: A pair name-value associated.
- Scalability: An algorithm is scalable if its running time grows (linearly) in proportion to the data set size.